A VERILOG-BASED SIMULATION METHODOLOGY FOR ESTIMATING POWER AND AREA

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

In

VLSI Design and Embedded Systems

By

RAMESH GUNTUPALLI

Roll No: 209EC2126

Department of Electronics and Communication Engineering
National Institute of Technology
Rourkela-769008
2011

A VERILOG-BASED SIMULATION METHODOLOGY FOR ESTIMATING POWER AND AREA

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

In

VLSI Design and Embedded Systems

By

RAMESH GUNTUPALLI

Roll No: 209EC2126

Under the Guidance of

Prof. KAMALAKANTA MAHAPATRA

Department of Electronics and Communication Engineering
National Institute of Technology
Rourkela-769008
2011

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

CERTIFICATE

This is to certify that the thesis report entitled **"A VERILOG-BASED SIMULATION METHODOLOGY FOR ESTIMATING POWER AND AREA"** submitted by **Mr.RAMESH GUNTUPALLI**, **Roll No: 209EC2126**, in partial fulfillment of the requirements for the award of Master of Technology degree in **Electronics and Communication Engineering Department** with specialization in **"VLSI Design and Embedded Systems"** at the **National Institute of Technology, Rourkela** is an authentic work under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Place**:** NIT Rourkela

Date:

Prof. K.K.Mahapatra

(Supervisor)

Dept. of Electronics & Communication Engg.

National Institute of Technology,

Rourkela - 769008.

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people (especially my family) who supported me and believed in me.

I express my deep sense of gratitude to **Dr. K. K. Mahapatra,** Professor in the department of Electronics and Communication Engineering, NIT Rourkela for giving me the opportunity to work under him and lending every support at every stage of this project work. I am indebted to his esteemed guidance, constant encouragement and fruitful suggestions from the beginning to the end of this thesis. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him.

I am thankful to all my teachers **Prof. S.K.Patra, Prof. G.S. Rath, Prof. S.Meher, Prof. D.P.Acharya** and **prof. N.V.L.N.Murthy** for providing a solid background for my studies and research thereafter.

I would like to thank my parents, brother, friends, seniors of VLSI lab-I and classmates who always encouraged me in the successful completion of my thesis work.

Ramesh Guntupalli

Roll No: 209EC2126

Abstract

Accurate modeling and estimating of the power dissipation in the early stages of the design flow is becoming more important, as the aggressive scaling of transistors results in higher leakage currents. New and complex systems are being implemented using highly advanced Electronic Design Automation (EDA) tools. As the complexity increases, the dissipation of power has emerged as one of the very significant design constraints. Low power designs are not only used in small size applications like cell phones and handheld devices but also in high-performance computing applications. Numerous tools have emerged in recent years to address this issue of power consumption and power optimization. With a vast number of these power measurement tools emerging, analyzing power consumed by digital circuits has not only become easier but also more effective methods are deployed to optimize digital circuits to dissipate less power.

In this work, we present a Verilog-based technique to estimate an accurate power dissipation of a design considering the state-dependency of the leakage power and path dependency of dynamic power. We develop the verilog models of cells which trace the probability of the static levels of the signals in the course of a simulation. Then, these data are used to calculate the power dissipation in the overall design. The power dissipation of some benchmark circuits is estimated using the proposed approach.

Contents

List of Figures

List of Tables

List of Acronyms

1. **EDA** - Electronic Design Automation
2. **VLSI** - Very Large-Scale Integration
3. **CMOS** - Complementary Metal Oxide Semiconductor
4. **RTL** - Register Transfer Level
5. **VCS** - Verilog Compiled Simulator
6. **HDL** - Hardware Design Language
7. **SDF** - Standard Delay Format
8. **SAIF** - Switching Activity Interchange Format
9. **VCD** - Value Change Dump
10. **VHDL** - Very High Speed IC Hardware Description Language
11. **SPICE** - Simulation Program with Integrated Circuit Emphasis
12. **TCL** - Tool Command Language

CHAPTER

1 Introduction

1.1 Overview of the Problem

With the increase in speed, mobility and miniaturization of current electronic products, the power consumption of these products has become a major design factor. Especially for mobile devices, the power consumption determines the battery life-time, the generated heat and the required heat dispersion measures. Therefore, the designers and consumers of electronic devices, as well as environmental considerations, demand a reduction in the power dissipation of digital circuits.

Digital circuit consists of a number of interconnected logic gates which together perform a function on one of more input signals. Every time an input signal changes, the change propagates via the gates through the circuit, causing signal switching activity in every place where the signal propagates to. This signal switching activity causes a current to charge or discharge the capacitive load of CMOS gates, which results in power dissipation. This power dissipation depends on the CMOS fabrication technology, operating frequency, but most of all on the switching activity per clock cycle within the digital circuit.

The increasing in usage of hand-held wireless devices and Internet appliances, there is a corresponding increased need for employing low-power design methodologies. One of the important requirements to know during a design process is how much power the circuit should dissipate considering its application. So after the designer writes the required code, keeping in mind all the specifications that have been given to him, a power calculation needs to be done to confirm if the design meets the required specification.

This is done prior to sending the chip for fabrication. So it is extremely important to get accurate power values using power determining tools running them at certain input conditions.

Numerous EDA (Electronic Design Automation) tools have been developed to not only determine power but also help in power reduction. Some of these tools are targeted pacifically for use in the power domain. The usage of these tools is classified depending on the layer of abstraction they are used in. The three main layers of abstraction include the RTL (Register Transfer Level), the gate and the transistor level. Though there are numerous tools that can be used at each of these levels, this thesis mainly concentrates on using Synopsys tools. The various power values that can be calculated using one of the tools is given in brief in the next section with detailed information following in the subsequent chapter.

1.2 Literature Review

A. Nourivand, Chunyan Wang and M. Omair Ahmad [1], have proposed Accurate modeling and estimating of the leakage power dissipation in the early stages of the design flow is becoming more important, as the aggressive scaling of transistors results in higher leakage currents. In this work, they present a VHDL-based technique to estimate an accurate leakage power of a design considering the state-dependency of the leakage power. They develop the VHDL models of cells which trace the probability of the static levels of the signals in the course of a simulation. Then, these data are used to calculate the leakage power in the overall design. The leakage power of some benchmark circuits is estimated using the proposed approach and the results are compared with those obtained from SPICE simulation, in order to illustrate the viability of the proposed technique. It is shown that the values of the leakage power obtained by the proposed technique are comparable to those obtained by SPICE, with a reduction of about three orders of magnitude in the simulation time.

They have proposed a VHDL-based technique for dynamic and leakage power estimation of combinational gate-level circuits. Integrating simulation and power estimation into an environment is useful for an improved utilization of VHDL for the power critical deep- submicron VLSI systems design. In this approach, they have developed power models of cells which trace the state probability as well as the transition probability of the signals in the course of a simulation. These data are later used to accurately estimate the state-dependent leakage and path-dependent dynamic power dissipation of the design. It is demonstrated that the proposed scheme can achieve accuracy comparable to that of SPICE in leakage power estimation, with about three orders of magnitude speedup in simulation time. The results also show that the leakage power contribution to the total power dissipation is not significant for this particular 0.18 μm technology. Therefore, the high accuracy offered by the proposed technique is more desirable for more advanced technologies which have considerable leakage currents.

Yibin Ye, Shekhar Borkar and Vivek De [4] have proposed a new standby leakage control technique, which exploits the leakage reduction offered by transistor stacks with "more than one 'off' device", demonstrates 2X reduction in standby leakage power for a 32-bit static CMOS adder in a low-Vt, sub-IV, and 0.1 μm technology. Leakage reduction is achieved with minimal overheads in area, power and process technology. The dynamics of leakage reduction due to transistor stacks, and its influence on the overall leakage power of large circuits are elucidated for the first time.

They demonstrated a new standby leakage control technique, which exploits the leakage reduction offered by transistor stacks with "more than one 'off'device". Up to 2X reduction in standby leakage power can be achieved by this technique with minimal overheads in area, power and process technology. We also elucidate the dynamics of leakage reduction due to transistor stacks, and its influence on overall leakage power of large circuits.

Vivek De and Shekhar Borkar [8] discussed key barriers to continued scaling of supply voltage and technology for microprocessors to achieve low-power and high-

performance. In particular, they focus on short-channel effects, device parameter variations, excessive subthreshold and gate oxide leakage, as the main obstacles dictated by fundamental device physics. Functionality of special circuits in the presence of high leakage, SRAM cell stability, bit line delay scaling, and power consumption in clocks & interconnects, will be the primary design challenges in the future. Soft error rate control and power delivery pose additional challenges. All of these problems are further compounded by the rapidly escalating complexity of microprocessor designs. The excessive leakage problem is particularly severe for battery-operated, high-performance microprocessors.

This paper has evaluated past trends in technology. It shows that trends in performance, density, and power have followed the scaling theory. If these trends continue, then power delivery and dissipation will be the biggest limiters. To overcome these limiters, die size growth will have to be constrained, and supply voltage scaling will have to continue. The threshold voltage will have to scale to meet the performance demand, resulting in higher subthreshold leakage current, limiting functionality of special circuits, increasing leakage power, soft error susceptibility, short channel effects, and device parameter variations. These are some of the major challenges that circuit designers will face in the future technologies.

A.Sagahyroon, J. Placer, M. Burmood and Mehran Massoumi [10], have proposed that recently, power dissipation has become a major design constraint for complex VLSI circuits. Designers need tools that rapidly, but accurately, estimate power dissipation in a given design. Two categories of tools are useful for this purpose: one is power optimization tools and algorithms tightly integrated with logic optimization, and second is an analysis tool for estimating the power consumption in an existing netlist. This work addresses the latter issue by employing a VHDL-based approach for analysis of power consumption in static CMOS combinational logic designs. The circuits under test will be either the result of logic synthesis with various optimization constraints or hand designs done through schematic capture. The proposed approach will also be used to analyze various known architectures of the same network for power consumption, such as various

forms of adders. The work presented in this article consists of three phases: (1) Designing smart VHDL simulation models that first measure transition activity at each node of the netlist and then estimate the power based on this activity and on fanout at each node, (2) the generation of smart input stimuli that achieve an upper bound on transition activity and hence power consumption, and third is an analysis of different topologies of the same circuit. The estimates produced by this analysis may provide useful feedback to designers or synthesis tools, allowing for better exploration of the design space.

Incorporating power-estimation techniques within VHDL is an appreciable step towards the utilization of VHDL as the basis for an integrated design environment for VLSI circuits. A critical issue in trying to estimate maximum power dissipation in CMOS circuits is that power is input-pattern dependent. Hence, the number of simulations that must be performed in order to find the maximum power dissipation is exponential in the number of inputs to circuit. In this work they proposed and made use of smart stimuli generated by utilizing genetic algorithms to develop smart test benches that tend to maximize the switching activity in structural VHDL models. In some cases the maximizing vectors have succeeded in producing the maximum possible activity in an economical CPU time. In addition, different topologies for the same network were compared for power consumption. Certain topologies exhibited superior power savings compared to others. Glitching has persistently contributed approximately 30% of the switching activity, making it a primary concern when designing for reduced power.

1.3 Out Line of the Thesis

Chapter 2 covers the basic concepts of the work. Chapter 3 discusses the tools used for the work. Chapter 4 gives experimental design. Chapter 5 presents experimental results. Chapter 6 shows the conclusion and future work.

CHAPTER

2 Basic Concepts of Power Dissipation

2.1 Need for Low Power Design

In the early 1970's designing digital circuits for high speed and minimum area were the main design constraints. Most of the EDA tools were designed specifically to meet these criteria. Power consumption was also a part of the design process but not very visible. The reduction of area of digital circuits is not as big issue today because with new IC production techniques, many millions of transistors can be fit in a single IC. However, shrinking sizes of circuits have paved the way for reduced power consumption in order to have an extended battery life. Also in submicron technologies, there is a limitation on the proper functioning of circuits due to heat generated by power dissipation. Market forces are demanding low power for not only better life but also reliability, portability, performance, cost and time to market. This is very true in the field of personal computing devices, wireless communications systems, home entertainment systems, which are becoming popular now-a-days. Devices that are also used for high-performance computing particularly need to dissipate less power to function correctly and for a long period of time.

Keeping all these in mind, low power design has become one of the most important design parameters for VLSI (Very Large Scale Integration) systems.

2.2 Sources of Power Dissipation

Generally, power is consumed when capacitors in the circuits are either charged or discharged due to switching activities. So at higher levels of a system this power dissipation is conserved by reducing the switching activities which is done by shutting

down portions of the system when they are not needed. Large VLSI circuits contain different components like a processor, a functional unit and controllers. The idea of power reduction is to stop any of the components of the processor when they are not needed so that less power will be dissipated when the processor is operating.

The power dissipation of digital CMOS circuits can be described by

$$Pavg = P\ dynamic + P\ short\text{-}circuit + P\ leakage + P\ static$$

Pavg is the average power dissipation, P dynamic is the dynamic power dissipation due to switching of transistors, P short-circuit is the short-circuit current power dissipation when there is a direct current path from power supply down to ground , P leakage is the power dissipation due to leakage currents, P static and is the static power dissipation[2]. Fig.1 describes the different components of power dissipation.

2.2.1 Static Power

Static power is the power dissipated by a gate when it is not switching that is, when it is inactive or static. Ideally, CMOS (Complementary Metal Oxide Semiconductor) circuits dissipate no static (DC) power since in the steady state there is no direct path from Vdd to ground. This scenario can never be realized in practice, since in reality the MOS transistor is not a perfect switch. There will always be leakage currents [2], sub threshold currents, and substrate injection currents, which give rise to the static component of power dissipation. The largest percentage of static power results from source-to-drain sub threshold voltage, which is caused by reduced threshold voltages that prevent the gate from completely turning off.

The diode leakage occurs when a transistor is turned off and another active transistor charges up or down the drain with respect to the first transistor's bulk potential. The resulting current is proportional to the area of the drain diffusion and the leakage current density. The diode leakage is typically 1 Pico A for a 1 micro-meter minimum feature size! The sub threshold leakage current for long channel devices increases linearly with the ratio of the channel width over channel length and decreases exponentially with $V_{GS} - Vt$ where V_{GS} is the gate bias and Vt is the threshold voltage. Several hundred

millivolts of “off bias” (say, 300-400 *mV*) typically reduce the sub threshold current to negligible values. With reduced power supply and device threshold voltages, the sub threshold current will however become more pronounced. In addition, at short channel lengths, the sub threshold current also becomes exponentially dependent on drain voltage instead of being independent of V_{DS}. The sub threshold current will remain 102 - 105 times smaller than the “on current” even at submicron device sizes.

2.2.2 Dynamic Power

Dynamic power is the power dissipated when the circuit is active. A circuit is active anytime the voltage on net changes due to some stimulus applied to the circuit. In other words, dynamic power dissipation is caused by the charging. Because voltage on an input net can change without necessarily resulting in logic transition in the output, dynamic power can be dissipated even when an output net doesn’t change its logic state. This component of dynamic power dissipation is the result of charging and discharging parasitic capacitances in the circuit.

Dynamic power of a circuit is composed of

a) Switching power

b) Internal power

2.2.2.1 Switching power

The switching power of a driving cell is the power dissipated by the charging and discharging of the load capacitance at the output of the cell. The total load capacitance at the output of a driving cell is the sum of the net and gate capacitances on the driving output. The charging and discharging are result of logic transitions. Switching power increases as logic transitions increase. Therefore, the switching power [2] of a cell is a function of both the total load capacitance at the cell output and the rate of logic transitions. Switching power comprises 70-90 percent of the power dissipation of an active CMOS circuit.

Dynamic power consumption depends linearly on the physical capacitance being switched [2]. So, in addition to operating at low voltages, minimizing capacitances offers

another technique for minimizing power consumption. In order to consider this possibility we must first understand what factors contribute to the physical capacitance of a circuit.

Power dissipation is dependent on the physical capacitances seen by individual gates in the circuit. Estimating this capacitance at the behavioral or logical levels of abstraction is difficult and imprecise as it requires estimation of the load capacitances from structures which are not yet mapped to gates in a cell library; this calculation can however be done easily after technology mapping by using the logic and delay information from the library.

Interconnect plays an increasing role in determining the total chip area, delay and power dissipation, and hence, must be accounted for as early as possible during the design process. The interconnect capacitance estimation is however a difficult task even after technology mapping due to lack of detailed place and route information. Approximate estimates can be obtained by using information derived from a companion placement solution or by using stochastic / procedural interconnect models. Interconnect capacitance estimation after layout is straight-forward and in general accurate.

2.2.2.2 Internal power

Internal power is any power dissipated within the boundary of a cell. During switching, a circuit dissipates internal power by the charging or discharging of any existing capacitances internal to the cell. Internal power includes power dissipated by a momentary short circuit between the P and N transistors of a gate, called short-circuit power. In most simple library cells, internal power is due mostly to short-circuit power. Library developers can model internal power by using the internal power library group [2]. The short-circuit (crowbar current) power consumption for an inverter gate is proportional to the gain of the inverter, the cubic power of supply voltage minus device threshold, the input rise/fall time, and the operating frequency. The maximum short circuit current flows when there is no load; this current decreases with the load. If gate sizes are selected so that the input and output Rise/fall times are about equal, the short-

circuit power consumption will be less than 15% of the dynamic power consumption. If, however, design for high performance is taken to the extreme where large gates are used to drive relatively small loads, then there will be a stiff penalty in terms of short-circuit power consumption.

2.2.3 Short-Circuit Power

The short-circuit power consumption, P short-circuit, is caused by the current flow through the direct path existing between the power supply and the ground during the transition phase.

2.2.4 Leakage Power

The nMOS and PMOS transistors used in a CMOS logic circuit commonly have non-zero reverse leakage and sub-threshold currents. These currents can contribute to the total power dissipation even when the transistors are not performing any switching action. The leakage power dissipation, P leakage is caused by two types of leakage currents

a) Reverse-bias diode leakage current

b) Sub threshold current through a turned-off transistor channel.

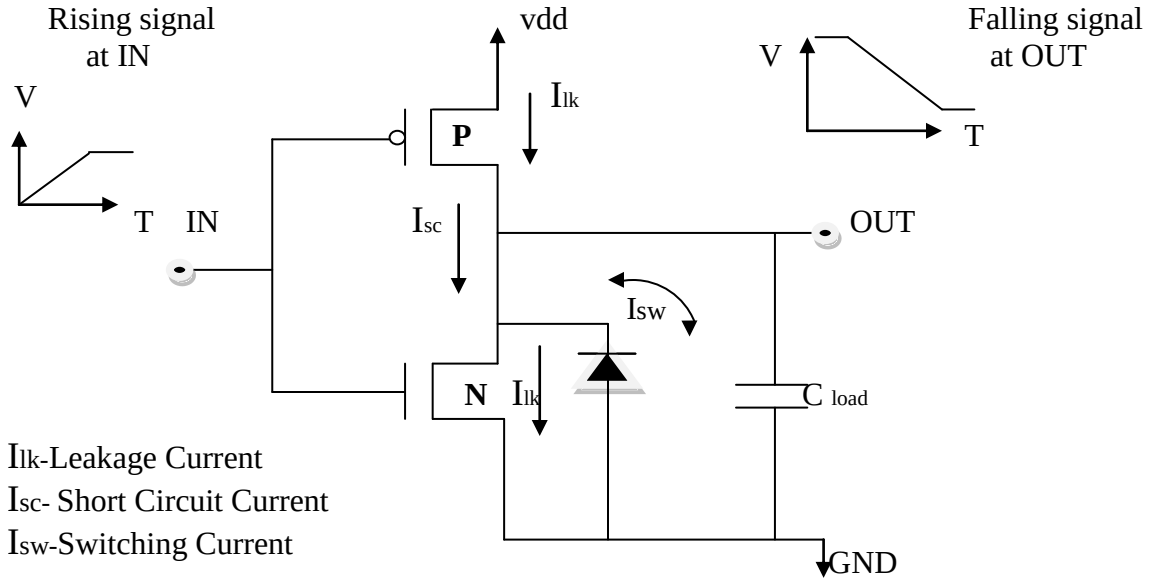

Fig.1 Different components of power dissipation.

CHAPTER

3 Tools

There has been a variety of tools involved in this thesis. Even though, this thesis is all about power calculations of circuits which are done using tools; there are other tools that have been used prior to the usage of power tools to give the required input to the power tools. More emphasis is given to these tools that are mainly involved in power estimation. The usage of tools has been classified as Power tools and Non-Power tools.

3.1. Non Power Tools

Non-power tools include Simulation tools, Synthesis tools and Waveform viewers. The tools that are discussed in this chapter are some of the non-power tools involved in the entire design flow. A short description of each of these tools along with their working flow is given in this chapter to understand their functionality. The subsequent chapter discusses each of the power tools in detailed manner as most of the thesis involves the use of these power tools. The following chapter also discusses the design flow from code writing to spice net-list simulation, clearly explaining the usage of these tools at the respective level.

3.1.1. Simulation Tool

Initially, to start with the Verilog or VHDL code for a particular design is written and tested. Simulation is done using Mentor's Modelsim for both VHDL and Verilog or other Verilog simulators. Modelsim is a simulation and a debugging tool for VHDL, Verilog, and other mixed-language designs from Mentor Graphics. The basic simulation

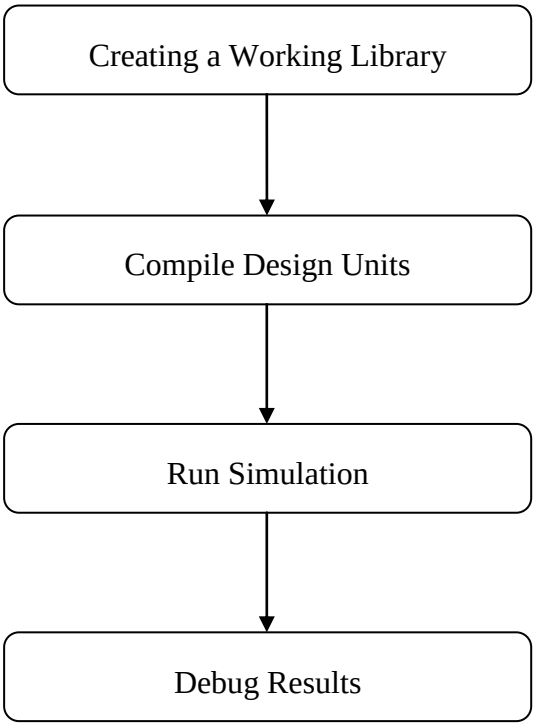

Fig.3.1 Modelsim simulation flow

flow is as shown in Figure 3.1. To start with a working library is created and the code is compiled using the commands depending upon whether the code is VHDL or Verilog. Verilog Compiled Simulator (VCS) from Synopsys is a high-performance, high-capacity Verilog simulator that incorporates advanced high-level abstraction, verification into an open platform.

The basic work flow for VCS [15] consists of two basic steps:

a) Compiling source files into executable binary files

b) Running the executable binary file

This two-step approach simulates the design faster and uses less memory than other interpretive simulators. The basic design flow is given in Figure 3.2.

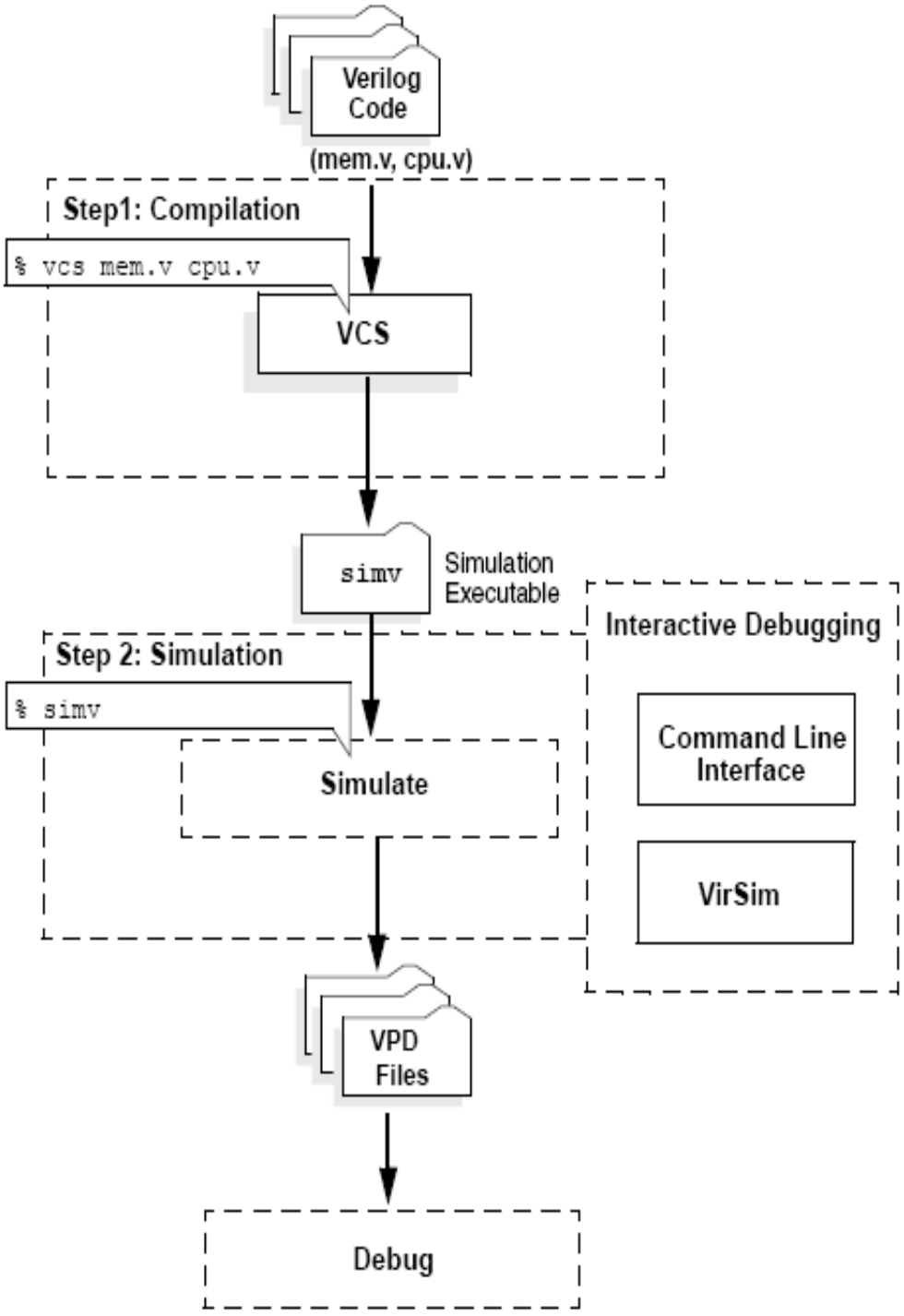

Fig 3.2 VCS work flow

3.1.2. Synthesis Tool

Design Compiler [15] is the core of the Synopsys synthesis software products. It comprises tools that synthesize HDL designs into optimized technology-dependent, gate-level designs. It supports a wide range of flat and hierarchical design styles and can optimize both combinational and sequential designs for speed, area, and power.

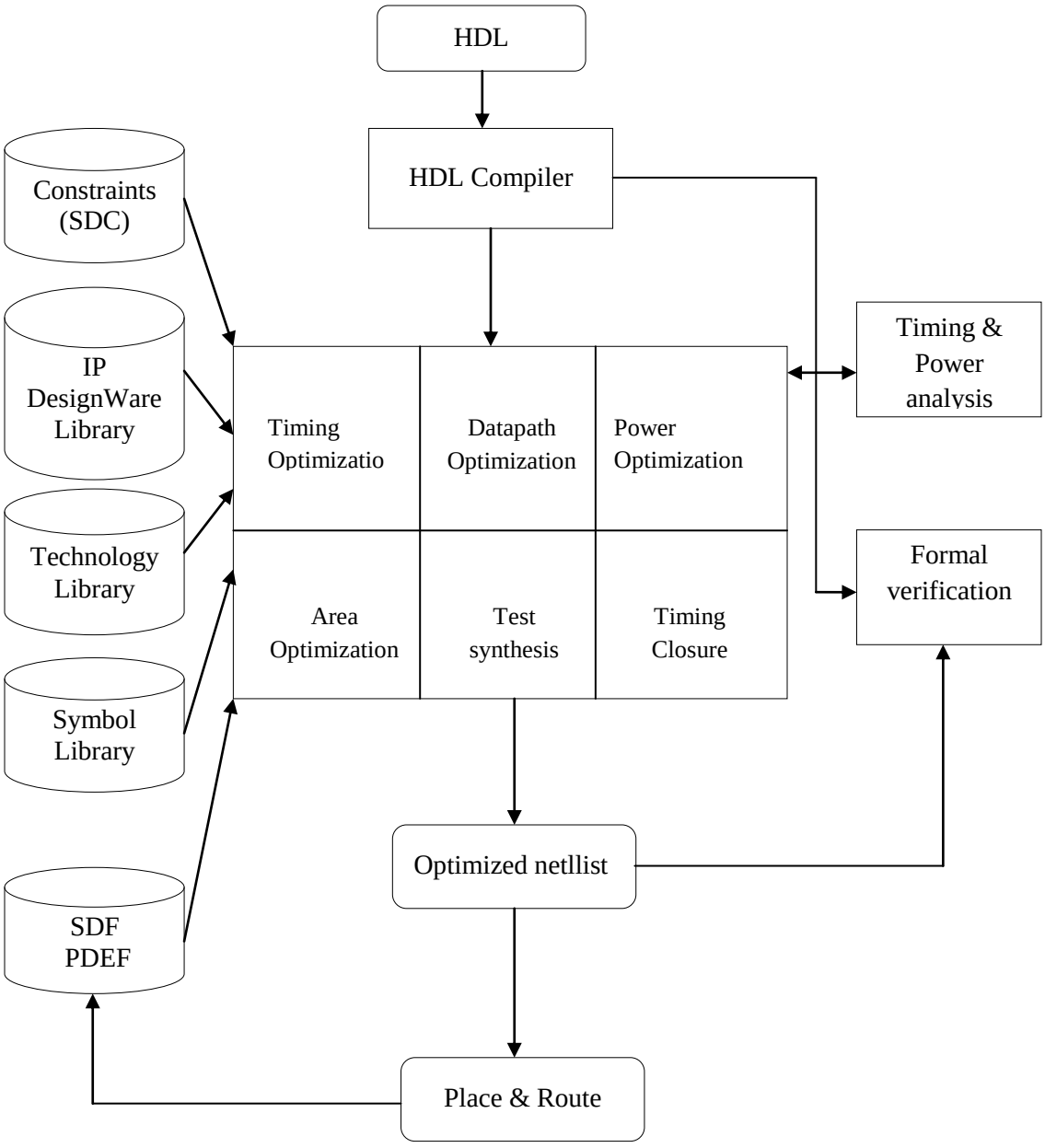

Fig.3.3 Design Compiler and Design Flow

Design Compiler reads and writes design files in all the standard EDA formats, including Synopsys internal database (.db) and equation (.eqn) formats. In addition, Design Compiler provides links to EDA tools, such as place and route tools, and to post-layout resynthesis techniques, such as in-place optimization. Design Compiler products include DC Professional, DC Expert, DFT Compiler, DC Ultra, and DC Ultra Plus.

The basic Design Compiler [15] design flow is given in Figure 3.3.

You use Design Compiler for logic synthesis, which is the process of converting a design description written in a hardware description language such as Verilog or VHDL into an optimized gate-level netlist mapped to a specific technology library. The steps in the synthesis process are as follows:

1. The input design files for Design Compiler are often written using a hardware description language (HDL) such as Verilog or VHDL.

2. Design Compiler uses technology libraries, synthetic or DesignWare libraries, and symbol libraries to implement synthesis and to display synthesis results graphically.

During the synthesis process, Design Compiler translates the HDL description to components extracted from the generic technology (GTECH) library and DesignWare library. The GTECH library consists of basic logic gates and flip-flops. The DesignWare library contains more complex cells such as adders and comparators. Both the GTECH and DesignWare libraries are technology independent, that is, they are not mapped to a specific technology library. Design Compiler uses the symbol library to generate the design schematic.

3. After translating the HDL description to gates, Design Compiler optimizes and maps the design to a specific technology library, known as the target library. The process is constraint driven. Constraints are the designer's specification of timing and environmental restrictions under which synthesis is to be performed.

4. After the design is optimized, it is ready for test synthesis. Test synthesis is the process by which designers can integrate test logic into a design during logic synthesis. Test synthesis enables designers to ensure that a design is testable and resolve any test issues early in the design cycle. The result of the logic synthesis process is an optimized gate-level netlist, which is a list of circuit elements and their interconnections.

5. After test synthesis, the design is ready for the place and route tools, which place and interconnect cells in the design. Based on the physical routing, the designer can back-annotate the design with actual interconnect delays; Design Compiler can then resynthesize the design for more accurate timing analysis.

3.2. Power Tools

This thesis involves the usage of Synopsys power tools. The power products are tools that comprise a complete methodology for low-power design. Synopsys power tools offer power analysis and optimization throughout the design cycle, from RTL to the gate level. Analyzing power early in the design cycle can significantly affect the quality of the design. Improvements made to the design while it is at RTL level can get even better results eventually. Not only these power tools do accurate measurements but also can help in calculating power quicker.

3.2.1. Power Compiler

Power Compiler [14] is an add-on product to Design Compiler. The Power Compiler tool optimizes the design for power. Working in conjunction with the Design Compiler tool, Power Compiler provides simultaneous optimization for timing, power and area. In addition to the standard inputs to synthesis (RTL or gate-level net-list, technology library, design constraints, and parasitics), Power Compiler uses two other inputs: Switching activity of design elements and power constraints. It contains all the analysis capabilities of DesignPower.

Power Compiler uses the same power analysis engine as Design Power. This allows Power Compiler to the use the same switching activity for optimization that Design Power uses for analysis. It accepts either user-defined switching activity, switching activity from simulation, or a combination of both. It provides RTL clock gating and optimizes the circuit based on circuit activity, capacitance, and transition times. Power Compiler cannot only be used as a standalone product but also can be used in coordination with Design Compiler, Module Compiler, Physical Compiler and Floor plan Manager.

3.2.3.1. Power Compiler Methodology

Power Compiler [14] is used at RTL and Gate level to calculate power and do power optimization depending on the need. At each level of abstraction, simulation, analysis and optimization can be performed to refine the design before moving to the next lower level. Simulation and the resultant switching activity gives the analysis and optimization the necessary information to refine the design before going to next lower level of abstraction. The higher the level of design abstraction, the greater the power savings can be achieved. The following Figure3.4 describes the power flow at each of the abstraction level. Figure 3.5 shows power flow from RTL to Gate level.

Cell internal power and net toggling directly affect dynamic power of a design. To report or optimize power, Power Compiler requires toggle information for the design. This toggle information is called Switching Activity.

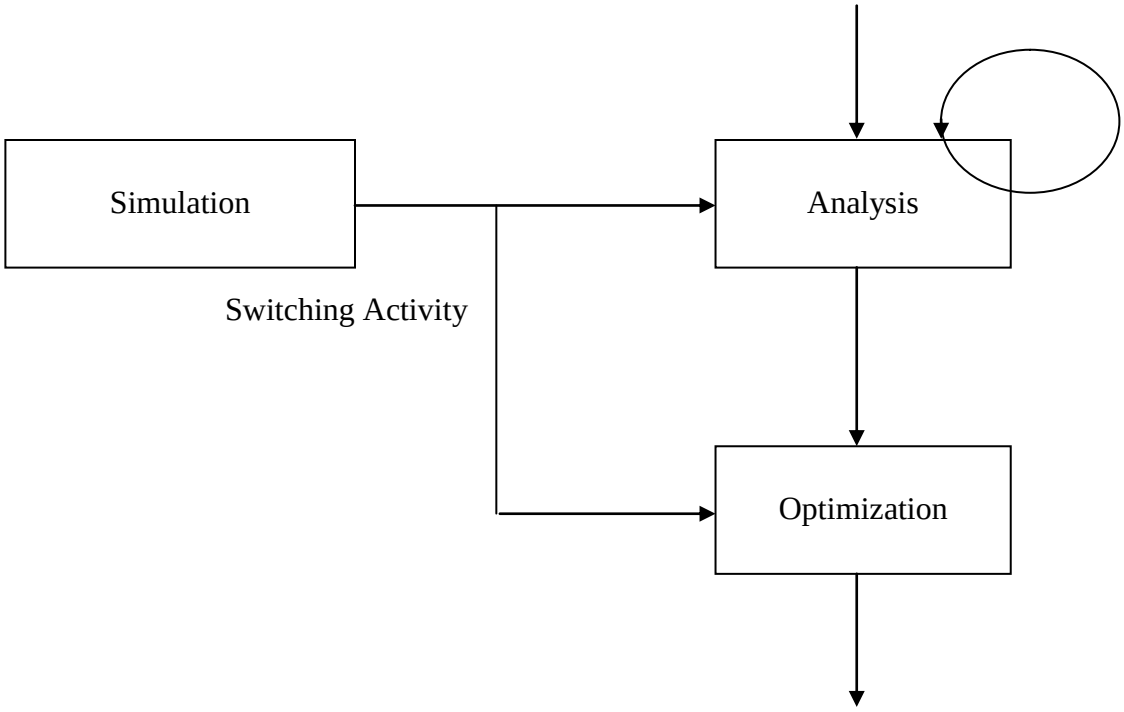

Fig.3.4 Power flow at each of the abstraction level

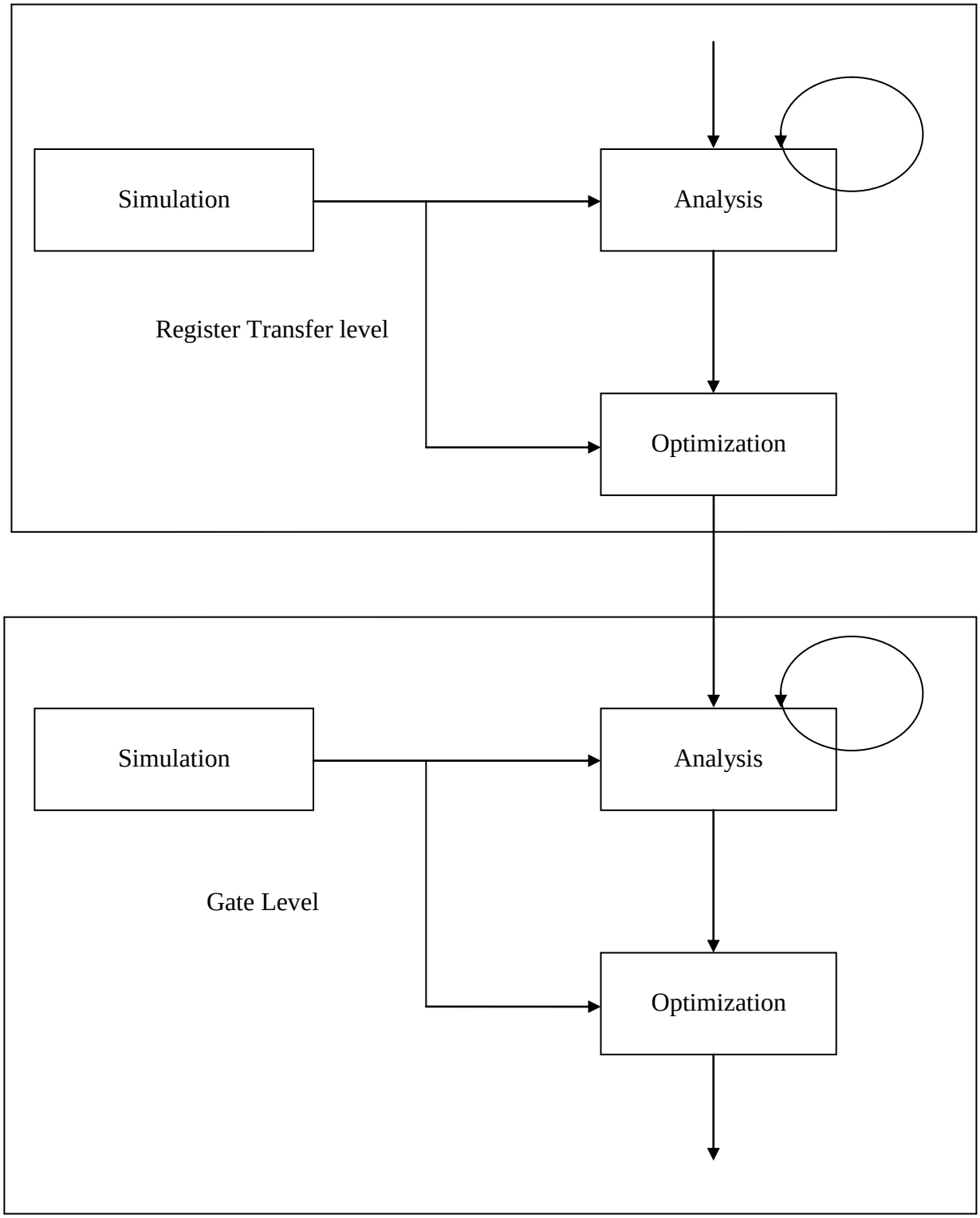

Fig.3.5 Power flow from RTL to Gate level

Power Compiler models switching activity in terms of static probability and toggle rate. Static probability is the probability that a signal is at a certain logic state and is expressed as a number between 0 and 1. It is calculated during simulation of the design by comparing the time of a signal at a certain logic state to the total time of the simulation. Toggle rate is the number of logic-0-to-logic-1 and logic-1-to-logic-0 transitions of a design object per unit of time.

CHAPTER

4 Experimental Design

The following ISCAS benchmark circuits [3] were tested as part of a research project. The research work involved was to create benchmark circuits for best PA (Power and Area) by exploring the best possible approach to improve those parameters. This thesis involves calculating the power of these circuits.

The following are the ISCAS benchmark circuits used in this experiment

a).ISCAS 85 Circuits

1) C432 (27-channel interrupt controller).

2) C499 (32-bit Single-Error-Correcting Circuit).

3) C880 (8-bit ALU).

4) C1908 (16-bit error detector/corrector).

5) C2670 (12-bit ALU and controller).

6) C3540 (8-bit ALU with binary and BCD arithmetic, and logic and shift operations).

7) C5315 (9-bit ALU).

8) C6288 (16x16 Multiplier).

b).74x Series Circuits

1) 74181 (4-Bit ALU/Function).

2) 74182 (carry look ahead circuit).

3) 74283 (Fast Adder Circuit).

4) 74L85 (4-Bit Magnitude Comparator).

4.1. Basic Synthesis Flow

Figure 4.1 shows the basic synthesis flow [15]. You can use this synthesis flow in both the design exploration and design implementation stages of the high-level design flow discussed previously.

Also listed in Figure 4.1 are the basic dc_shell commands that are commonly used in each step of the basic flow. For example, the commands analyze, elaborate, and read file are used in the step that reads design files into memory. All the commands shown in Figure 4.1 can take options, but no options are shown in the figure.
The basic synthesis flow consists of the following steps:

1. Develop HDL Files

The input design files for Design Compiler are often written using a hardware description language (HDL) such as Verilog or VHDL. These design descriptions need to be written carefully to achieve the best synthesis results possible. When writing HDL code, you need to consider design data management, design partitioning, and your HDL coding style. Partitioning and coding style directly affect the synthesis and optimization processes.
Note:
This step is included in the flow, but it is not actually a Design Compiler step. You do not create HDL files with the Design Compiler tools.

2. Specify Libraries

You specify the link, target, symbol, and synthetic libraries for Design Compiler by using the link_library, target_library, symbol_library, and synthetic_library commands. The link and target libraries are technology libraries that define the semiconductor

vendor's set of cells and related information, such as cell names, cell pin names, delay arcs, pin loading, design rules, and operating conditions. The symbol library defines the symbols for schematic viewing of the design. You need this library if you intend to use the Design Vision GUI. In addition, you must specify any specially licensed DesignWare libraries by using the synthetic_library command. (You do not need to specify the standard Design Ware library.)

3. Read Design

Design Compiler can read both RTL designs and gate-level netlists. Design Compiler uses HDL Compiler to read Verilog and VHDL RTL designs. It has a specialized netlist reader for reading Verilog and VHDL gate-level netlists. The specialized netlist reader reads netlists faster and uses less memory than HDL Compiler. Design Compiler provides the following ways to read design files:

- The analyze and elaborate commands
- The read_file command
- The read_vhdl and read_verilog commands.

These commands are derived from the read_file -format VHDL and read_file -format verilog commands.

4. Define Design Environment

Design Compiler requires that you model the environment of the design to be synthesized. This model comprises the external operating conditions (manufacturing process, temperature, and voltage), loads, drives, fanouts, and wire load models. It directly influences design synthesis and optimization results. You define the design environment by using the set commands listed under this step of Figure 4.1.

5. Set Design Constraints

Design Compiler uses design rules and optimization constraints to control the synthesis of the design. Design rules are provided in the vendor technology library to ensure that the product meets specifications and works as intended. Typical design rules constrain transition times (set_max_transition), fanout loads (set_max_fanout), and

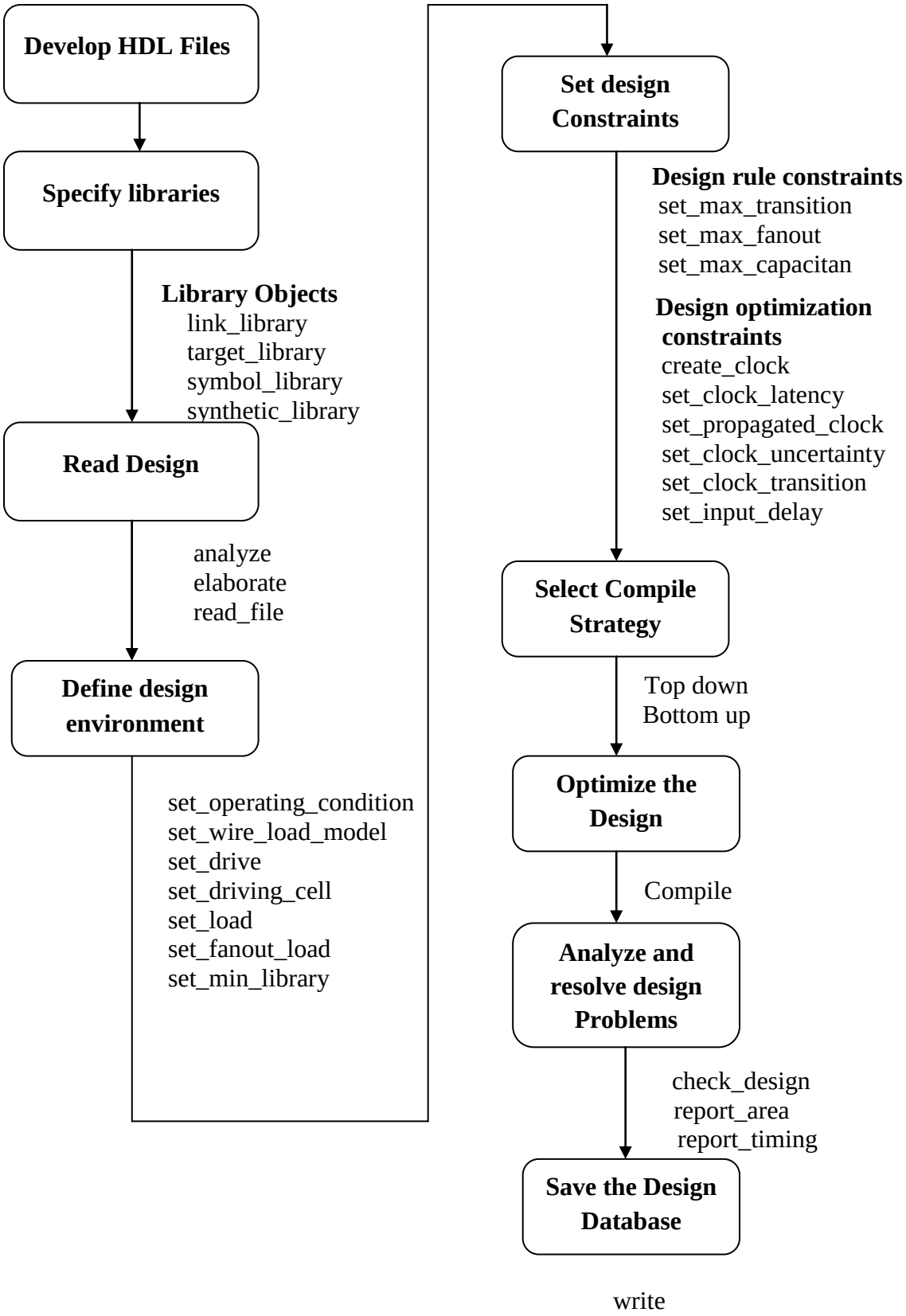

Fig 4.1 Basic Synthesis flow

capacitances (set_max_capacitance). These rules specify technology requirements that you cannot violate. (You can, however, specify stricter constraints.) Optimization constraints define the design goals for timing (clocks, clock skews, input delays, and output delays) and area (maximum area). In the optimization process, Design Compiler attempts to meet these goals, but no design rules are violated by the process. You define these constraints by using commands such as those listed under this step in Figure 4.1. To optimize a design correctly, you must set realistic constraints.

6. Select Compile Strategy

The two basic compile strategies that you can use to optimize hierarchical designs are referred to as top down and bottom up. In the top-down strategy, the top-level design and all its subdesigns are compiled together. All environment and constraint settings are defined with respect to the top-level design. Although this strategy automatically takes care of interblock dependencies, the method is not practical for large designs because all designs must reside in memory at the same time. In the bottom-up strategy, individual subdesigns are constrained and compiled separately. After successful compilation, the designs are assigned the dont_touch attribute to prevent further changes to them during subsequent compile phases. Then the compiled subdesigns are assembled to compose the designs of the next higher level of the hierarchy (any higher-level design can also incorporate unmapped logic), and these designs are compiled. This compilation process is continued up through the hierarchy until the top-level design is synthesized. This method lets you compile large designs because Design Compiler does not need to load all the uncompiled subdesigns into memory at the same time. At each stage, however, you must estimate the interblock constraints, and typically you must iterate the compilations, improving these estimates, until all subdesign interfaces are stable. Each strategy has its advantages and disadvantages, depending on your particular designs and design goals. You can use either strategy to process the entire design, or you can mix strategies, using the most appropriate strategy for each subdesign.

7. Optimize the Design

You use the compile command to invoke the Design Compiler synthesis and optimization processes. Several compile options are available. In particular, the map_effort option can be set to low, medium, or high. In a preliminary compile, when you want to get a quick idea of design area and performance, you set map_effort to low. In a default compile, when you are performing design exploration, you use the medium map_effort option. Because this option is the default, you do not need to specify map_effort in the compile command. In a final design implementation compile, you might want to set map_effort to high. You should use this option judiciously, however, because the resulting compile process is CPU intensive. Often setting map_effort to medium is sufficient.

8. Analyze and Resolve Design Problems

Design Compiler can generate numerous reports on the results of a design synthesis and optimization, for example, area, constraint, and timing reports. You use reports to analyze and resolve any design problems or to improve synthesis results. You can use the check_design command to check the synthesized design for consistency. Other check_ commands are available.

9. Save the Design Database

You use the write command to save the synthesized designs. Remember that Design Compiler does not automatically save designs before exiting. You can also save in a script file the design attributes and constraints used during synthesis. Script files are ideal for managing your design attributes and constraints.

4.2. Power Estimation Techniques

Power values for each of these circuits are done using power tools of Synopsys spread through two levels of abstraction, RTL level and Gate level. Power calculation for each of the tools at a specific level is done using a different methodology and with other non-power tools involved. The first method involves using Power Compiler with RTL level switching activity and the second method involves using Power Compiler with Gate

Level switching activity. The accuracy of the power values obtained using these tools gets better as we move from RTL level to gate level. This is because the information required for calculating accurate power of a circuit is given in more detail as the level goes to the lower levels of abstraction and also the tools involved get more complex at those levels. Finally, a table is made with power values filled for each of the circuit.

4.3. Power Estimation Methodology

The following Figure 4.2 shows the methodology [14] of power calculation using the combination of Power Compiler and Design Compiler. The flow of data between the different steps and tools used are also shown. Before starting to calculate power using Power Compiler the desired gate-level net-list of the design should be first generated. The power methodology starts with the RTL design and finishes with a power-optimized gate-level net-list. Ultimately, Power Compiler is used to calculate power using the gate-level net-list produced by the Design Compiler or power-optimized gate net-list produced by Power Compiler itself.

As seen in the figure most of the processes that take place are using Design Compiler, but the simulation process that is shown is outside Design Compiler tool and is done as part of power calculation. The main purpose of simulation is to generate information about the switching activity of the design and create a file called Back-annotation.

This file can contain switching activity from RTL simulation or gate-level simulation. Initially, the RTL design is given to the HDL compiler to create a technology-independent format called as GTECH design. This is as a result of analyzing and elaborating the design by HDL compiler. This formatted design is given as an input to Design Compiler. Before it is compiled by the Design Compiler, "**rtl2saif**" command is used to create forward-annotation file which is later used for simulation. The formatted design GTECH is later given as input to Design Compiler which produces an output which is given to Power Compiler.

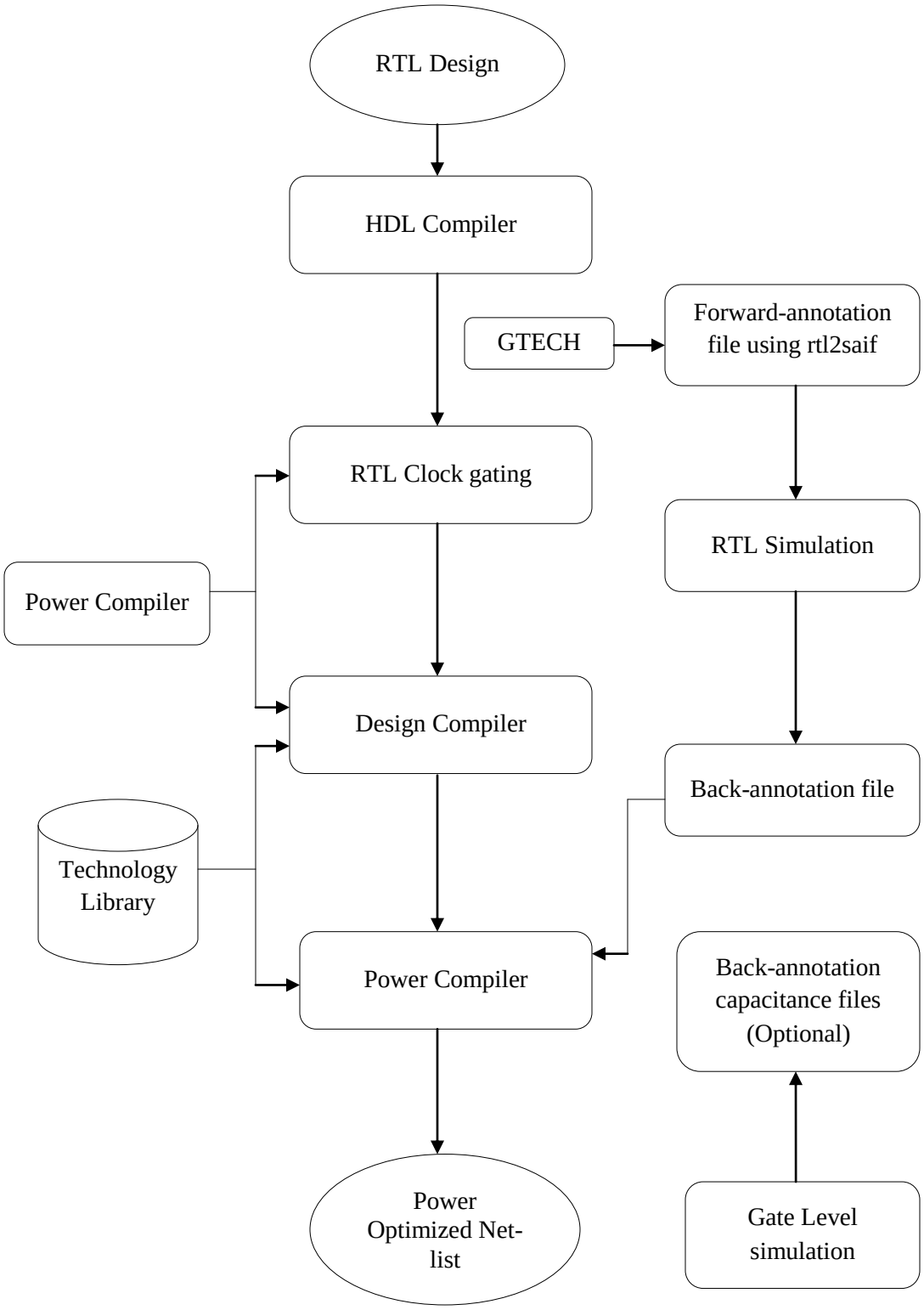

Fig.4.2 Power Estimation Methodology in Power Compiler

The Forward-annotation SAIF file is given as an input to do RTL simulation which gives a back-annotation SAIF file which is used by Power Compiler. This forward annotated file contains directives that determine which design elements to be traced during simulation. Gate-level simulation can also use a library forward-annotation file.

This forward-annotation file used for gate level simulation has different information compared to RTL forward-annotation file. This file contains information from the technology library about cells with state and path-dependent power models. "**Lib2saif**" command is used to get this forward-annotation file.

During power analysis, Power Compiler uses the annotated switching activity to evaluate the power consumption of the design. During power optimization, Power Compiler uses the annotated switching activity to make decisions about the design.

4.3.1. Capturing Forward and Backward Switching Activity

Power Compiler [14] requires information about the switching activity of the design to do power analysis. The forward and back-annotation files are in SAIF format. SAIF is an ASCII format developed at Synopsys to facilitate the interchange of information between simulators and Synopsys power tools. Some of the power tools cannot understand SAIF file so in that case VCD file is used. Depending on the tool, either RTL level switching activity or Gate-level switching activity is used. Power Compiler has a methodology that enables the use of switching activity from RTL simulation as well as from Gate-level simulation. Using gate-level simulation the power values are much more accurate but doing that is time consuming. During RTL and gate level simulation the designer can direct the simulator to monitor and write out the switching activity of certain important elements in the design. For accurate analysis, synthesis-invariant elements should be closely monitored during RTL simulation. These are the elements that are not changed during simulation like primary inputs, sequential elements, black boxes, three-state devices and hierarchical ports.

4.4. Power Optimization

Power optimization achieved at higher levels of abstraction (RTL) has an impact in reducing power in the final gate-level optimization. Power Compiler performs clock gating when the design is elaborated using "**-gate_clock**" option. Design generally has synchronous load-enable registers. These registers are formed using feedback loops by Design Compiler.

These registers maintain the same logic value through multiple cycles and unnecessarily use power. When the "**-gate_clock**" option is used HDL compiler introduces gates in the clock network before Design Compiler does its processing. During the next step, Design Compiler checks the gated clock introduced by HDL compiler and uses simple registers without synchronous load-enabled functionality thus saving power. RTL clock gating is achieved without affecting timing or area of the design.

At the gate level, Design Compiler and Power Compiler are used to create gate-level net-list optimized for power. Once the RTL clock gating [14] is done, the next output is the gate-level net-list which will be optimized for power. First constraints are set for timing and area. Then the design is compiled using the Design Compiler. This creates a gate-level design on which the switching activity can be annotated using the back-annotation file. The back-annotation file is read into Power Compiler using "**read_saif**" command. After this power constraints are set to trigger power optimization by Power Compiler. Then the design is compiled using Power Compiler. Using the switching activity and power constraints, Power Compiler produces a gate-level net-list which is optimized for timing, power and area. Switching activity from RTL simulation provides good power optimization results. However, switching activity from gate level simulation provides much more accurate analysis and optimization. The power analysis of the gate-level design can be done at various points in the entire methodology. Once annotating the switching activity from the back-annotation file, power can be analyzed before compiling using Power Compiler. This is done before power optimization. Once doing power optimization the power values can be compared. "**report_power**" is the command used to get detailed power results.

CHAPTER

5 Experimental Results

This chapter gives details on the various results that have been obtained using the different circuits that were discussed earlier. Results are given in tables:

1). Power estimation results for ISCAS 85 bench mark circuits in different Conditions.

2). Power estimation results for 74x series benchmark circuits in different Conditions.

The following section discusses the different optimized circuits that are obtained using Design Vision and power estimation results obtained by using power Compiler.

1. Design Vision:

Design Vision is a GUI and an integrated part of the Design Compiler. The optimized gate level netlist is obtained by using Design Vision for all the benchmark circuits.

2. **Power Compiler:**

The power value is calculated using Power Compiler at the gate level level. The inputs to calculate power are Gate-level Net-list, RTL switching activity, obtained from Design Compiler.

5.1. ISCAS 85 benchmark Circuits

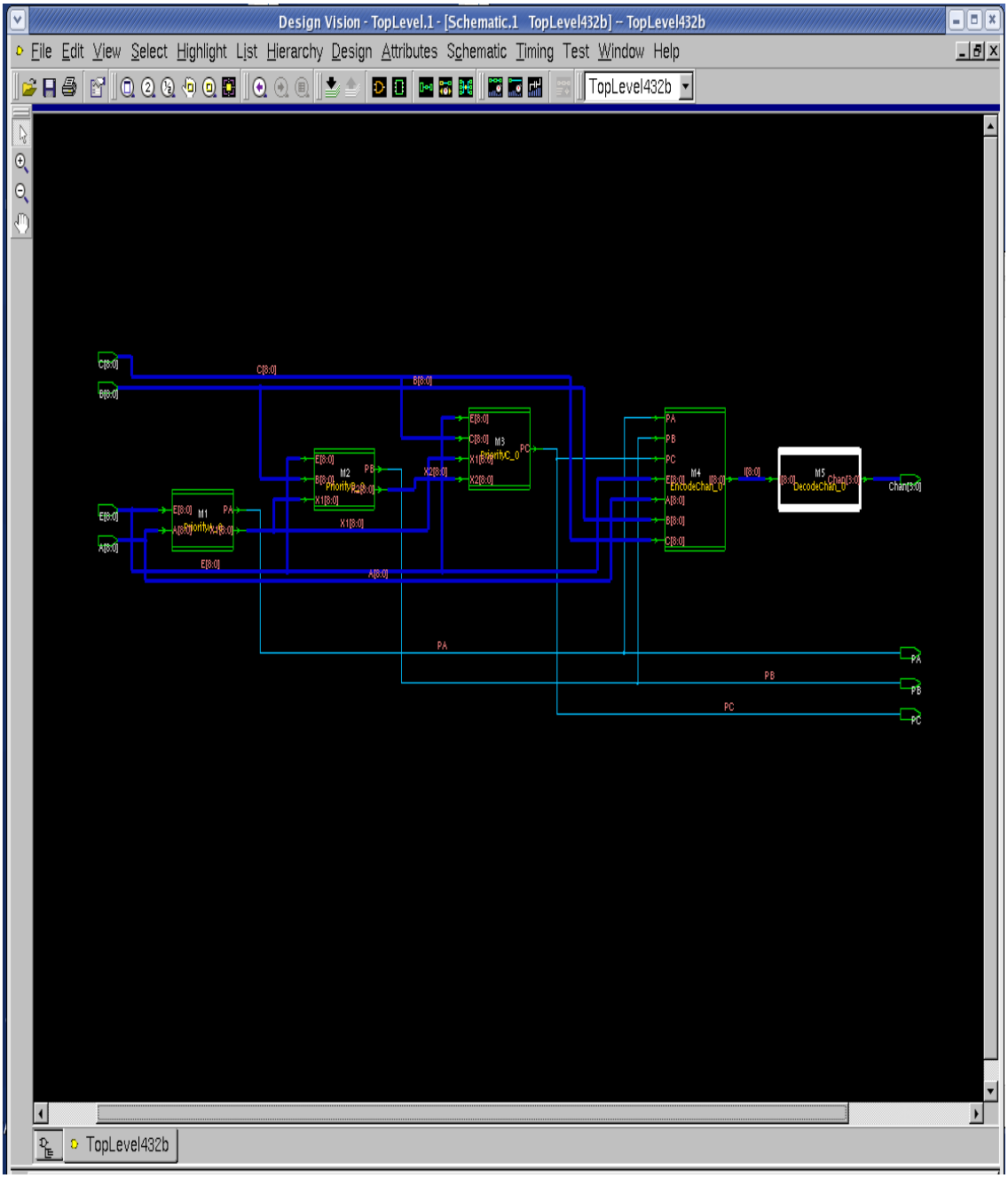

Fig.5.1 Optimized gate level netlist for C432

Fig.5.2 Optimized gate level netlist for C499

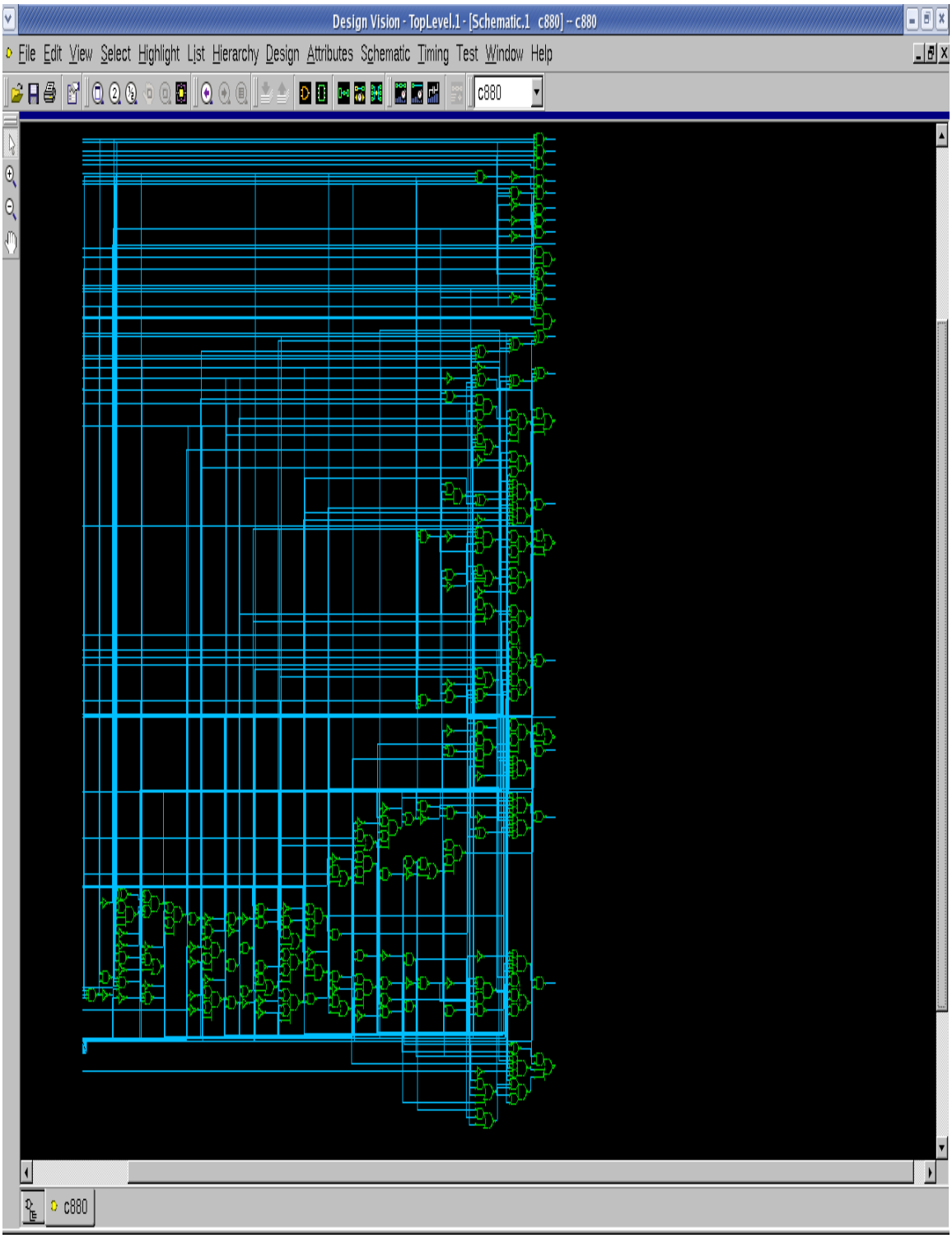

Fig.5.3 Optimized gate level netlist for C880

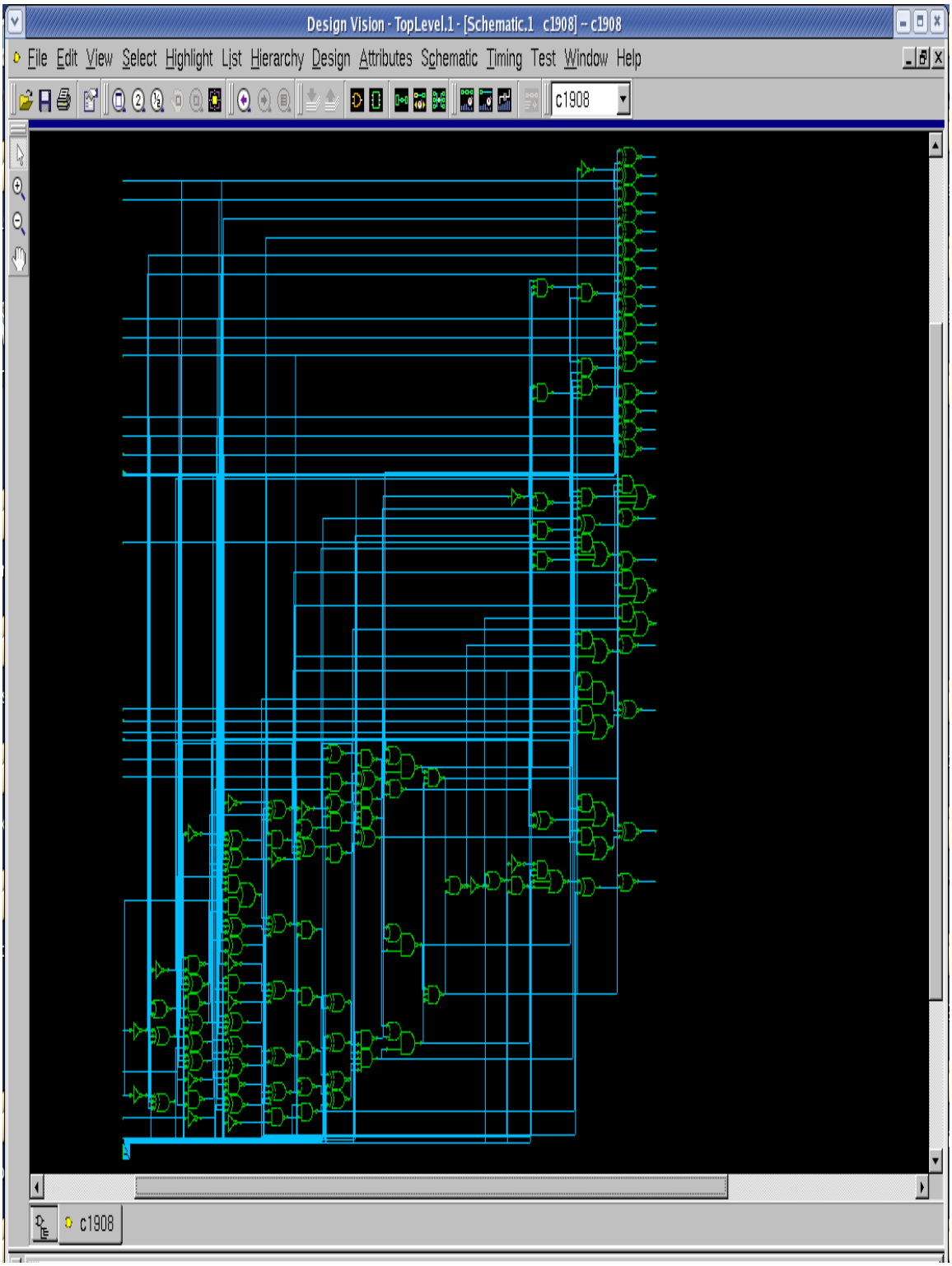

Fig.5.4 Optimized gate level netlist for C1908

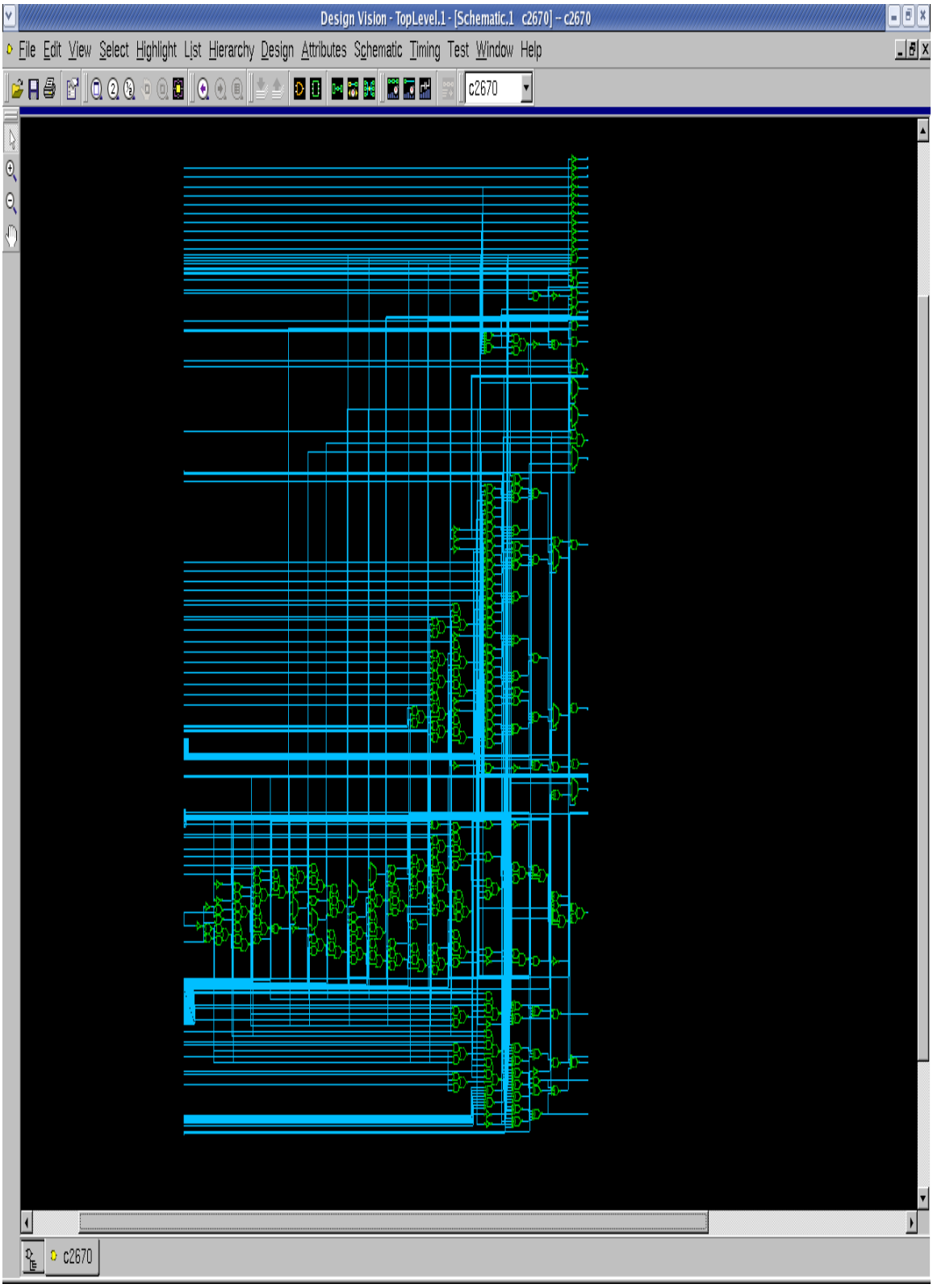

Fig.5.5 Optimized gate level netlist for C2670

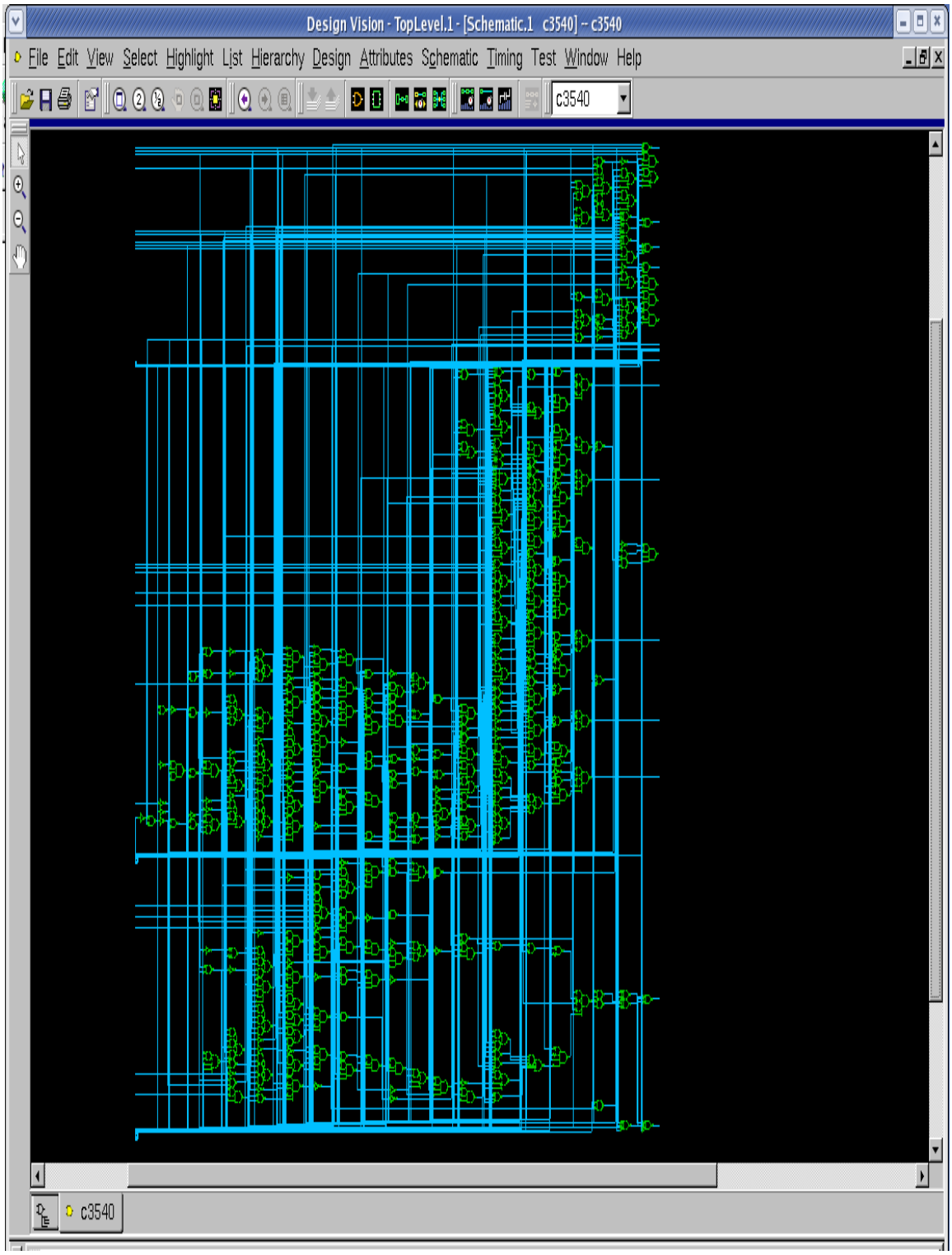

Fig.5.6 Optimized gate level netlist for C3540

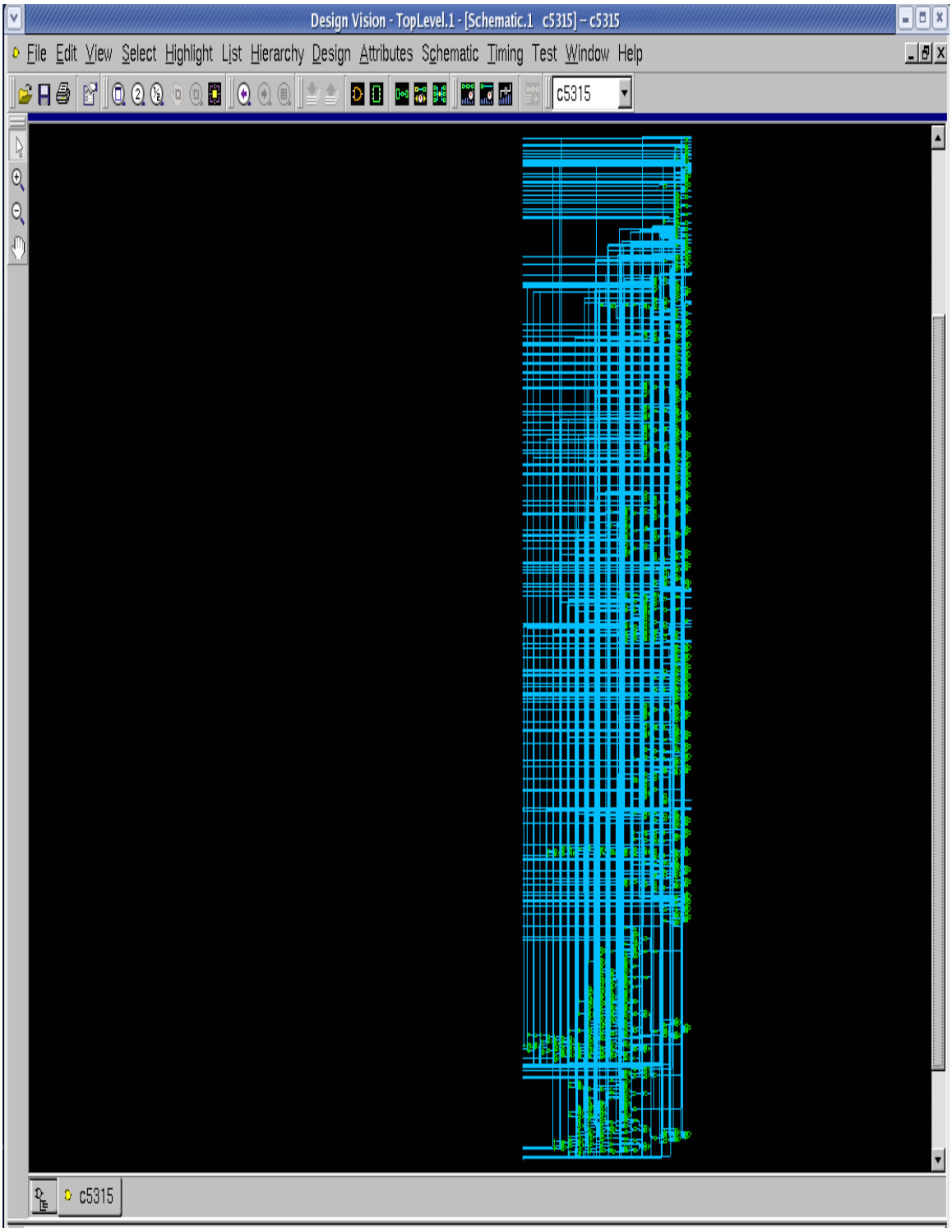

Fig.5.7 Optimized gate level netlist for C5313

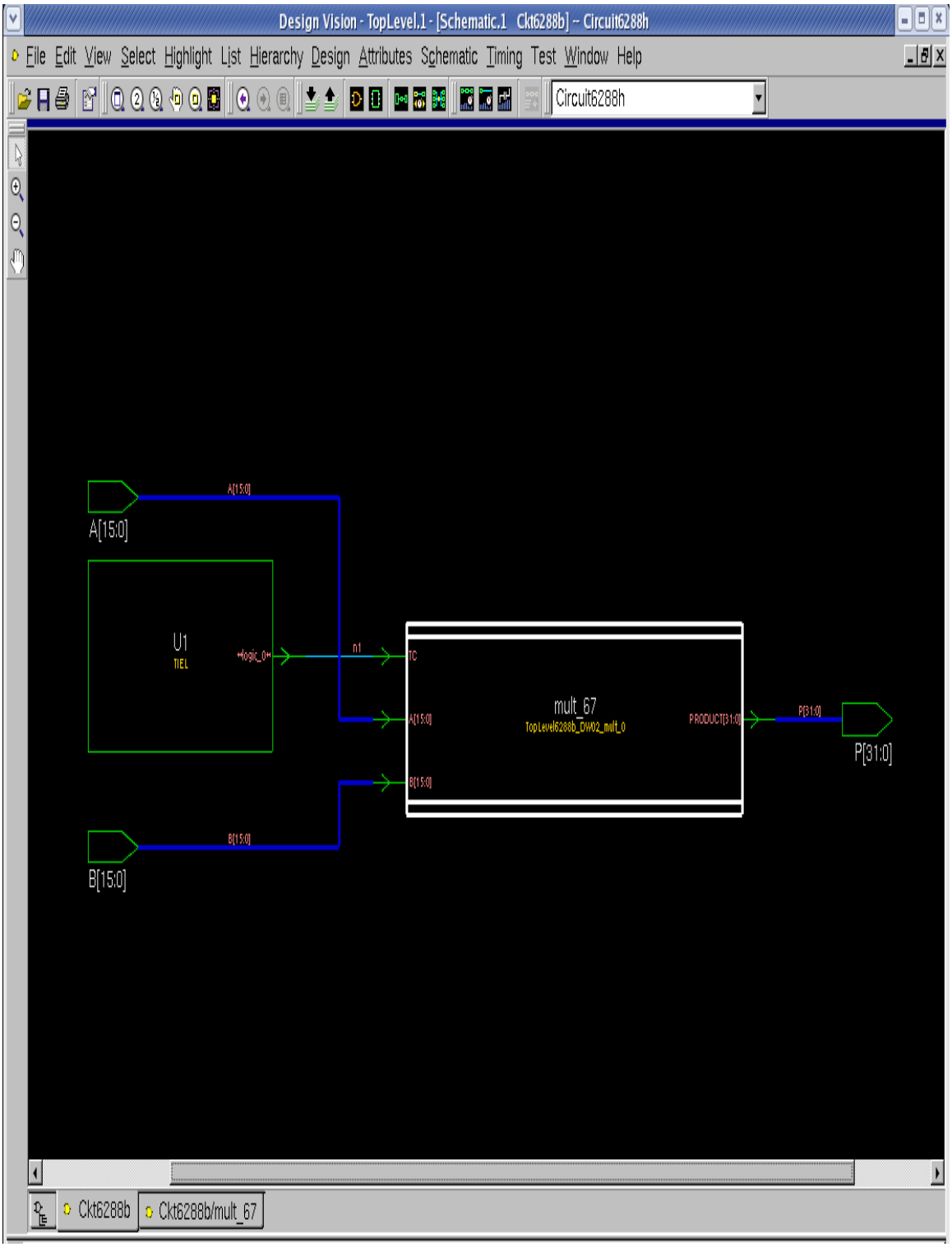

Fig.5.7 Optimized gate level netlist for C6288

Table 5.1 Power and Area Estimation Results for ISCAS benchmark in **Typical Corner**

Circuit	Function	Typical Corner		Area (μm^2)
		25^0 C		
		Leakage Power (nW)	Dynamic Power (µW)	
C432	27-Channel Interrupt Controller	654.85	15.94	199.08
C499	32-Bit Single-Error-Correcting Circuit	2368.00	53.51	448.55
C880	8-bit ALU	1187.20	33.92	379.44
C1908	16-bit error Detector/Corrector	2111.70	50.25	358.20
C2670	12-bit ALU and controller	3371.40	96.49	650.52
C3540	8-bit ALU with binary and BCD arithmetic, and logic and shift operations	3803.40	100.99	799.55
C5315	9-bit ALU	7028.10	213.28	1455.11
C6288	16X16 Multiplier	20705.0	512.06	2853.00

Table 5.2 Power and Area Estimation results for ISCAS benchmark in **Fast Corner**

Circuit	Function	Fast Corner		Area (μm^2)
		0^0 C		
		Leakage Power (nW)	Dynamic Power (µW)	
C432	27-Channel Interrupt Controller	4009.20	26.07	198.36
C499	32-Bit Single-Error-Correcting Circuit	1396.56	80.48	448.55
C880	8-bit ALU	7621.50	67.80	378.36
C1908	16-bit error Detector/Corrector	1021.51	67.27	354.60
C2670	12-bit ALU and controller	1560.96	150.82	656.28
C3540	8-bit ALU with binary and BCD arithmetic, and logic and shift operations	1550.50	162.11	802.79
C5315	9-bit ALU	3096.57	353.96	1457.64
C6288	16X16 Multiplier	8549.10	581.43	2624.39

5.2. 74x Series Circuits

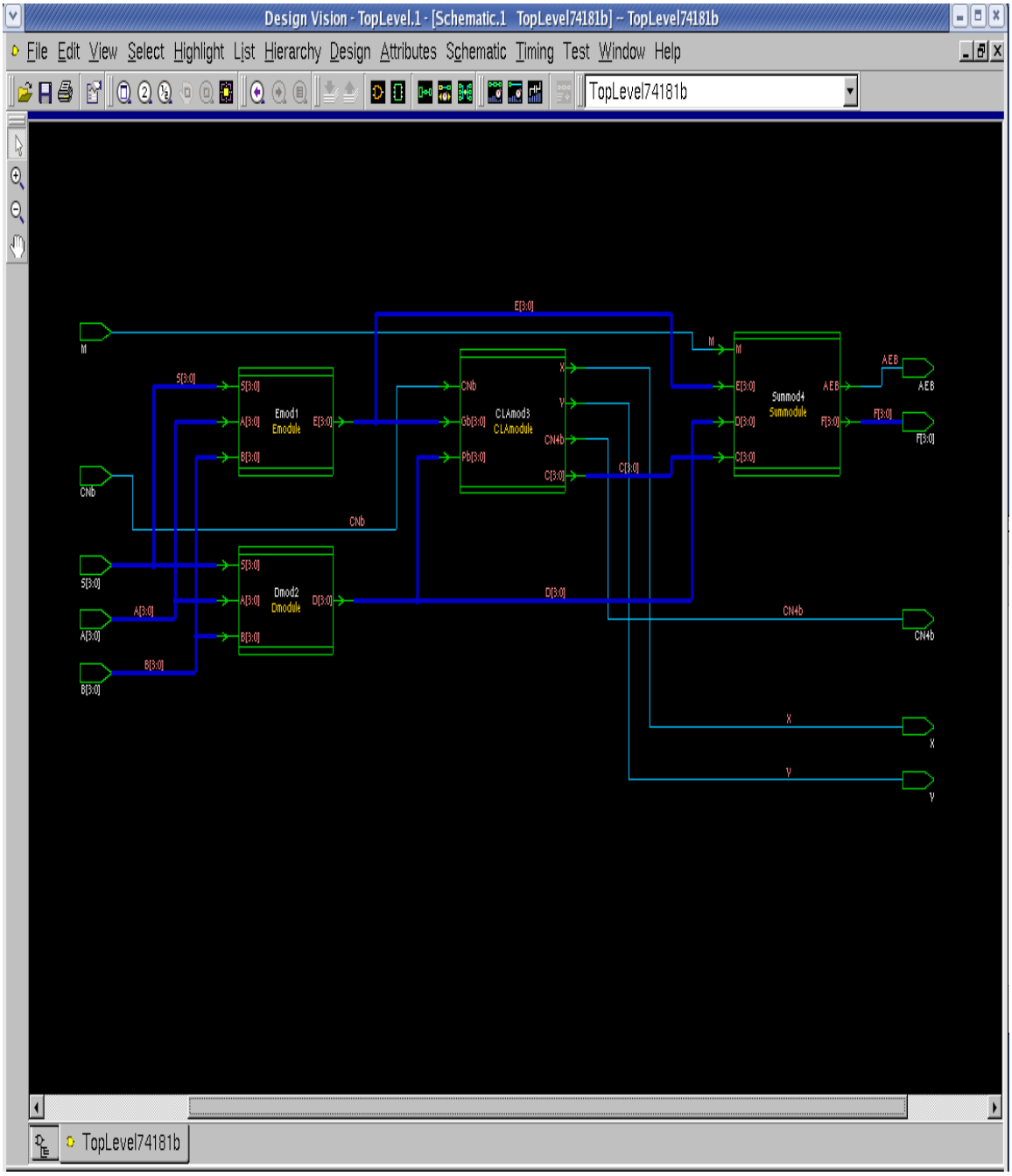

Fig.5.9 Optimized gate level netlist for 74181

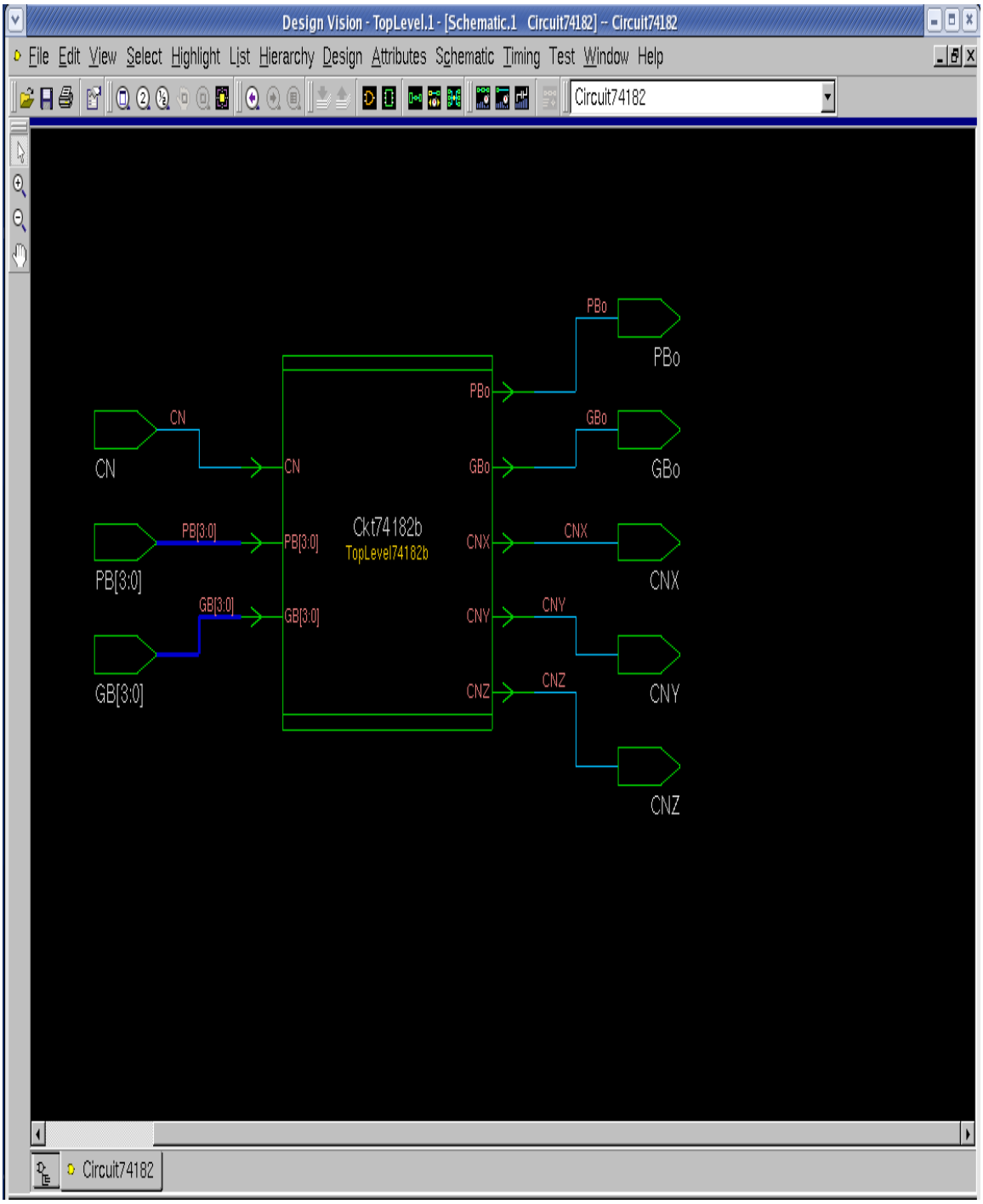

Fig.5.10 Optimized gate level netlist for 74182

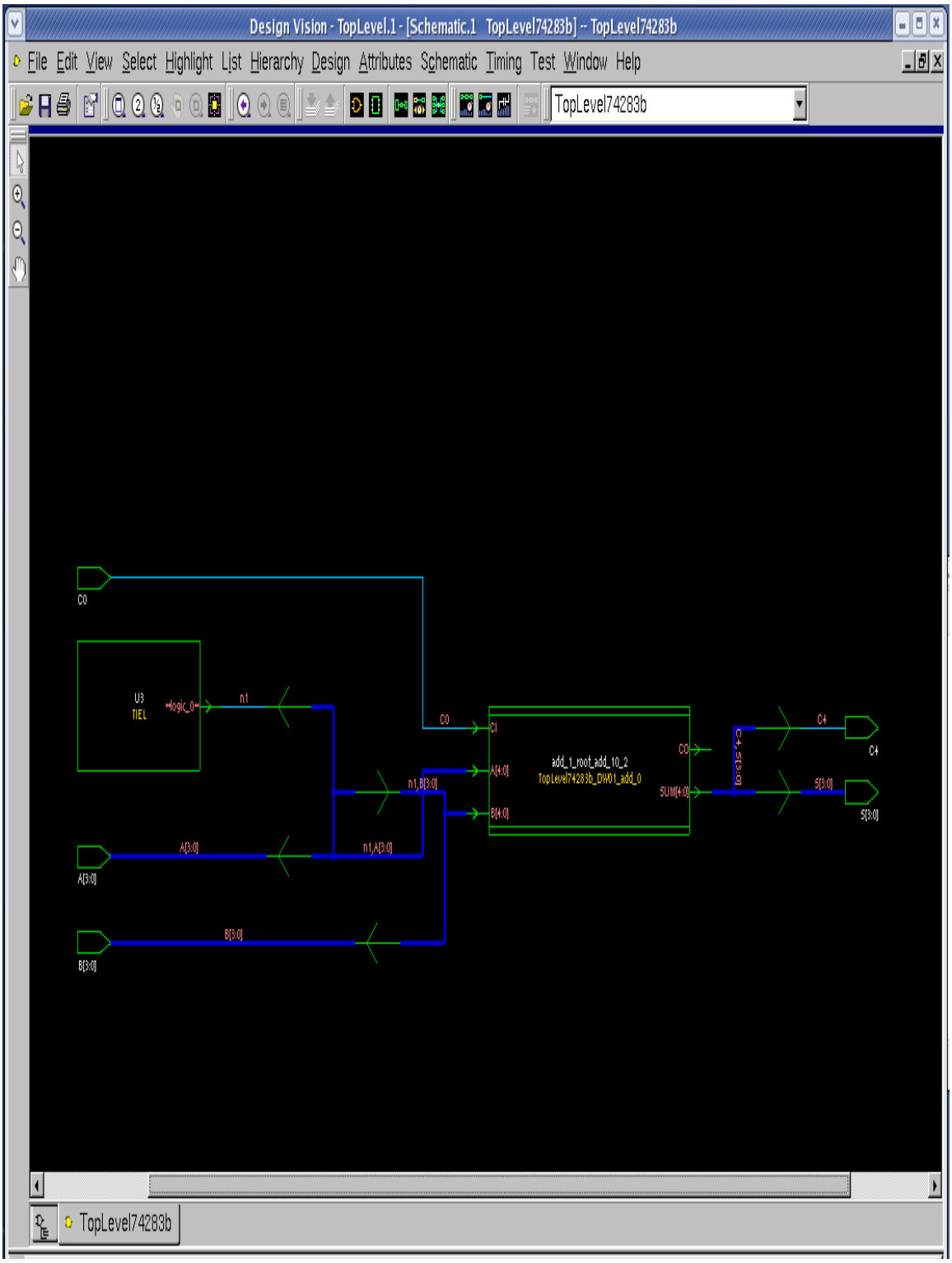

Fig.5.11 Optimized gate level netlist for 74283

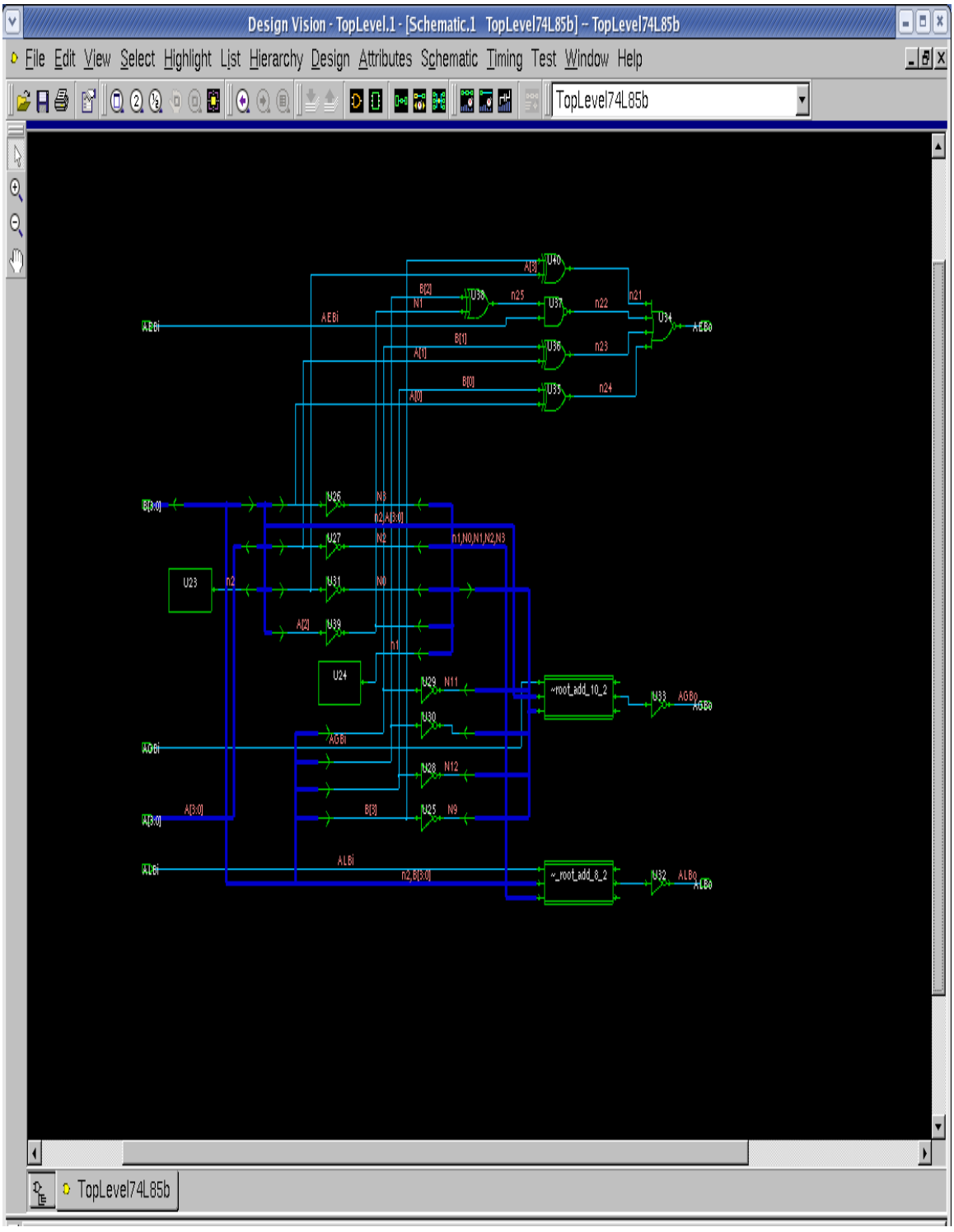

Fig.5.12 Optimized gate level netlist for 74L85

Table 5.3 Power and Area Estimation results for 74x Series Circuits in **Typical Corner**

Circuit	Function	Typical Corner		Area (μm^2)
		25^0 C		
		Leakage Power (nW)	Dynamic power (µW)	
74181	4-bit ALU	341.23	48.93	93.60
74182	4-bit CLA	59.60	1.95	20.16
74283	4-bit Adder	258.74	8.26	37.08
74L85	4-bit magnitude Comparator	637.09	8.26	105.48

Table 5.4 Power and Area Estimation Results for 74x series Circuits in **Fast Corner**

Circuit	Function	Fast Corner		Area (μm^2)
		0^0 C		
		Leakage Power (nW)	Dynamic power (µW)	
74181	4-bit ALU	1919.80	14.05	90.36
74182	4-bit CLA	345.81	9.63	21.24
74283	4-bit Adder	1341.90	8.96	37.08
74L85	4-bit magnitude Comparator	3609.50	10.76	105.84

CHAPTER

6 Conclusion and Future Work

6.1 Conclusion

A VHDL-based technique for dynamic and leakage power estimation of combinational gate-level circuits is proposed. Integrating simulation and power estimation into an environment is useful for an improved utilization of VHDL for the power critical deep-submicron VLSI systems design. In this approach, we have developed power models of cells which trace the state probability as well as the transition probability of the signals in the course of a simulation. These data are later used to accurately estimate the state-dependent leakage and path-dependent dynamic power dissipation of the design.

It can be concluded from these power estimations at different operating conditions of abstraction how inaccurate values at fast corner are compared to Typical Corner. The difference in the measurement can be seen in the tables for each of the circuit. Power estimation at that condition is done mainly because we can get faster results and can be used to decide on optimizing the circuit depending on the specification. The study proves that the methodology of calculating power is correct taking into example, benchmark circuit. There had been a lot of experiments conducted while performing power calculations using Power Compiler. It can be concluded from those experiments that proper usage of Power Compiler needs to be done keeping in the mind what circuit we are performing the simulations on, how much of accuracy is needed in the measurement and how fast we need the results.

Careful understanding of all these are to be deployed to get the best results from the High-Speed circuit simulator. Since the thesis is about power estimation, it would have been good to compare the values of circuits than best PA. PA circuits are developed so they have got the best Power and Area. For bigger circuits this value has proved to be more than their Default circuit. The power results obtained using Power Compiler using RTL Switching Activity , Power Compiler using Gate-Level Switching used power characterized library provided by TSMC 65nm technology library,. Power and Area being the three major constraints in designing digital circuits there are applications like tactical missile applications and other defense related projects that would require circuits to be kept in a smaller area, dissipate power and perform really fast. Keeping this mind checking power in a best PA circuit is afterall useful to be implemented in these devices.

Some other conclusions are since power values are dependent not only the circuits but also on the tools used, versions of the tool, power characterized library used, the input stimulus used and what the output load is, it would be only valid to compare results from different EDA power tools only if the above are identical. It would not be possible to get to any specific conclusion analyzing the results of Power Estimation other than trying to rewrite the RTL code (VHDL or Verilog) to get lesser power value as much as possible. The values obtained using Power Compiler are considered to be the true values since all the inputs given to the tool contain fine information about the circuits and it would be logical to compare it with the testing of the real chips in the lab. The power values obtained by the other tools are used only to select the best possible netlist that are capable of giving less power when realized into real chips.

6.2 Future Work

As seen in the conclusion, the real power comparison should have been between operating conditions of the circuits. With the flow of power estimation already developed as part of this work, Timing analysis for circuits can also be performed. Various values of power estimation can be reported. Placement and Routing of the circuits can be done using Synopsys tools as opposed to Cadence tools and power values obtained as the result of that can be compared. Other comparison like getting net-list using a different

extraction tool can also be done. With the use of sophisticated power measuring equipment, each of the circuit can be tested for power in real time for the same input vectors used here and values can be compared with the tool's estimation.

REFERENCES

[1] Nourivand.A, ChunyanWang, Ahmad.M.O "A VHDL-based technique for an accurate estimation of leakage power in digital CMOS circuits," in *the 3rd international IEEE-NEWCAS conf*., 2005.pp.47-50.

[2] D. Soudris, C. Piguet, and C. Goutis, "Designing CMOS Circuits for Low Power", Kluwer Academic Publishers, 2002.

[3] ISCAS bench mark circuits. www.eecs.umich.edu/~jhayes/**iscas**/-UnitedSataes

[4] "TSMC 65 nm core library application note," Release 1, Jun 2005. http://www.tsmc.com/

[5] V.De and S.Borkar "Technology and design challenges for low power and high performance," in *Int. Symp. Low Power Electronics and Design,* Aug.1999, pp.163-168.

[6] Chandrakasan et.al, "Design considerations and tools for low voltage digital system design," in *proc. 33rd, Design Automation Conference,* pp.113-118,199.

[7] J. Flynn, B. Waldo, "Power management in complex SoC design," http://www.synopsys.com/sps.

[8] Y. Ye, S. Borkar, and V. De, "A new technique for standby leakage reduction in high-performance circuits," in *Symp. VLSI Circuits Dig. Tech. Papers*, 1998, pp. 40–41.

[9] J. P. Halter and F. Najm, "A gate-level leakage power reduction method for ultra-low- power CMOS circuits," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1997, pp. 475–478.

[10] Sagahyroom.A, Placer.J, Burmood.M, Massoumi.M "A VHDL-based simulation methodology for estimating switching activity in static CMOS circuits*,"* in *proc. IEEE SAIC conf*. 1988, pp.295-300.

[11] J.Placer, A.Sagahyroon and M. Massoumi, "A Framework for Estimating Maximum Power Dissipation in CMOS Combinational Circuits Using Genetic Algorithms," *IEEE Southeastern Symposium on System Theory*, pp. 348-352, 1997.

[12] J.Y.Lin et al., "A Cell-Based Power Estimation in CMOS Combinational Circuits," *Proceedings of the International Conference on Computer-Aided-Design*, pp. 304-309, 1994.

[13] A.P. Chandrakasan et al., "Low-Power CMOS Digital Design," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 4, pp. 473-483, April 1992.

[14] Power Compiler user guide, Version Y-2006.06, June 2006.

[15] Design Compiler user guide, Version Y-2006.06, June 2006.

[16] VCS user guide, Version Y-2006.09, June 2006.

[17] Ashwin Balakrishnan, "An Experimental Study of the Accuracy of Multiple Power Estimation Methods", The University of Tennessee, Knoxville, MS thesis, August 2004.

[18] Synopsys's Power Compiler

www.synopsys.com/Tools/Implementation/.../**powercompiler**_ds.pdf